THE GEOSPATIAL METADATA SERVER: ON-LINE METADATA DATABASE, DATA CONVERSION AND GIS PROCESSING

Anup Pradhan

Doctorate in Philosophy

Department of Geography
University of Edinburgh

July 1999

EX BIBL. UNIV. EDINBURGH

DECLARATION

I declare that this dissertation represents my own work and that where the work of others is used it is duly accredited. I further declare that this dissertation consists of six chapters with a combined length of approximately seventy-five thousand words.

Anup Pradhan,

July, 1999

ACKNOWLEDGEMENTS

I would like to thank my first supervisor, Bruce Gittings, for his foresight in concluding a long time ago about the relevance of Internet technology towards GIS development and research. A number of key components for the Geospatial Metadata Server are, in fact, his initial ideas including the data-switch-yard, HTML forms/metadatabase connection, parts of the search engine and not to forget, drawing my attention to the freeware Fetch_HTTP utility. I would also like to thank Steve Dowers for his help with Pro*C, particularly where it is used for the on-line metadata parsing and database entry utility. Also, there were implementation problems associated with various components of GMS that I could not have overcome without the help of both Bruce and Steve. Thanks also goes to my second supervisor, Nick Hulton, for the knowledge, effort and patience he showed while reviewing the chapters of this dissertation. Thanks goes to Anu Mathew for helping me with the mathematics of the GMS search engine as well as the support of having a fellow Ph.D. student as one's comrade in arms. Finally, special thanks goes to my parents Mahesh and Kunj Pradhan for their faith in my ability to succeed in the completion of this dissertation and beyond.

ABSTRACT

The growth of the Internet and more specifically, the World Wide Web allows users, providers and site administrators of geospatial data to share information amongst each other. This helps minimise the time and costs associated with data collection and base mapping. The aim of this research is to better understand fundamental issues and software concepts that influence how geographical data is stored, catalogued and processed over the Internet. This is accomplished by, in part, developing an integrated system that merges existing technologies such as an HTTP server, a RDBMS and a GIS to facilitate the on-line search of standardised metadata along with the access and conversion of geospatial data. The prototype system developed to meet this aim is called the Geospatial Metadata Server (GMS).

The software prototype evaluates the problems and issues by suggesting strategies that might influence and improve the further implementation of on-line systems designed to share geographic information. The database uses standardised metadata, which enables on-line data dissemination sites to integrate indexing and querying capabilities amongst themselves and ensures that data providers and users share the same terminology for describing geospatial data. The use of an RDBMS substantially increases the speed and flexibility of query operations alongside the storage and report facilities for metadata. Because of the size of metadata records, the system provides the useful feature of parsing entire metadata files and loading the information directly into the database. Metadata files can also be manually entered, updated or deleted on-line in a way that divides the arduous task of entering text via forms into several separate parts. The amount of information contained within a metadata file that is necessary to represent appropriately one or more geospatial data sets is well beyond what the producer of the geospatial data is willing to enter into an on-line metadata catalogue in a single sitting.

Geospatial data occurs in a variety of different data formats. This is a fundamental cause why GIS software packages (supplied by different GIS vendors) fail to operate

between themselves. An easily accessible on-line data conversion utility for GMS was developed that is capable of inter-converting between pre-defined geospatial data formats. By examining the information retrieved from the database and feeding it to the server's software, the search engine integrates the metadata database and the conversion utilities. The resulting implications are that on-line GIS software is tailored to respond automatically to a Web search engine and that metadata, geospatial data and GIS software is able to exist separately on different physically remote servers. The implication is that Internet technology is becoming the basis for a new computing architecture for GIS by addressing the technical and institutional issues that arise in accessing and using on-line geographic information.

TABLE OF CONTENTS

DECLARATION II
ACKNOWLEDGEMENTS III
ABSTRACT IV
TABLE OF CONTENTS VI
LIST OF FIGURES XI
LIST OF TABLES XIX
LIST OF ACRONYMS XX

CHAPTER ONE 1

INTRODUCTION 1
1.1 THE AIM 1
1.2 RATIONALE 2
1.3 SPECIFIC OBJECTIVES 11
1.4 THESIS STRUCTURE 19

CHAPTER TWO 21

GIS IN THE INFORMATION AGE 21

2.1 METADATA AND INFORMATION DISCOVERY 22
2.1.1 Metadata standards 23
2.1.1.1 The CSDGM standard 23
2.1.1.2 The DIF standard 26
2.1.1.3 USMARC formats and Discovery metadata 27
2.1.2 Information discovery protocols 28

2.1.2.1 A system management protocol 29
2.1.2.2 A directory service protocol 29
2.1.2.3 A database search protocol 30
2.2 ON-LINE METADATA DATABASES AND CATALOGUES 32
2.2.1 Metadata storage and query 32
2.2.2 On-line metadata storage and query systems 35
2.3 METADATA PARSING PROGRAMS 44
2.4 METADATA CREATION TOOLS 46
2.4.1 Metadata editing programs 47
2.4.2 On-line metadata entry and modification programs 49
2.4.2.1 BIC Metadata Form 50
2.4.2.2 Metamorph 52
2.4.2.3 GCMD: WWW metadata entry form 54
2.5 INTEROPERABILITY IN GIS 58
2.5.1 Defining interoperability in GIS 58
2.5.2 Enhancing interoperability in GIS 60
2.5.3 Distributed GIS functionality 66
2.6 QUERYING GEOSPATIAL METADATA 68
2.6.1 Digimap interface 68
2.6.2 ScotLIS WWW interface 71
2.6.3 GCMD search-engine 73
2.7 WEB ROBOTS IN GIS 75
2.8 CHAPTER CONCLUSION 77

CHAPTER THREE 79

METADATA ENTRY AND STORAGE 79

3.1 THE GMS METADATABASE 80
3.1.1 Metadata standards 80
3.1.2 Software design 81

3.1.3 On-line metadata and relational databases *85*
3.1.4 GMS metadatabase implementation *87*
3.1.5 Design and implementation constraints *102*
3.2 METADATA PARSING AND DATABASE ENTRY 103
3.2.1 Software design *106*
3.2.2 Software implementation *110*
3.2.2.1 The user-interface and CGI script 110
3.2.2.2 The MPARSER utility 111
3.2.2.3 The MLOADER utility 116
3.2.3 Design and implementation constraints *125*
3.3 METADATABASE MAINTENANCE AND MANAGEMENT 127
3.3.1 The management and maintenance of on-line metadata *128*
3.3.2 Software design *130*
3.3.3 Software implementation *135*
3.3.3.1 The user-interface 135
3.3.3.2 The CGI script 141
3.3.3.3 The database entry programs 144
3.3.4 Design and implementation constraints *150*
3.4 CHAPTER CONCLUSION 151

CHAPTER FOUR 152

METADATA QUERY & ANAYLYSIS 152

4.1 AN INTEGRATED WWW SEARCH-ENGINE 153
4.1.1 Tools used to construct metadata search-engines *154*
4.1.2 Software design *157*
4.1.3 Software implementation *159*
4.1.3.1 The user-interface 159
4.1.3.2 The search-engine CGI script 162
4.1.3.3 The search-engine algorithm 166

4.1.4 Design and implementation constraints *176*
4.2 GEOSPATIAL DATA-CONVERSION UTILITIES 178
4.2.1 Geospatial data formats and data-conversion *178*
4.2.2 Software design *181*
4.2.3 Software implementation *190*
4.2.3.1 The user-interface 190
4.2.3.2 The DXFARC, DLGARC and TIGERARC conversion utilities 193
4.2.3.3 The ARCDXF, ARCDLG, ARCTIGER conversion utilities 201
4.2.3.4 The e00 file decompression and plotting utilities 204
4.2.4 Design and implementation constraints *207*
4.3 A GEOSPATIAL WEB ROBOT 208
4.3.1 Software design *209*
4.3.2 Software implementation *213*
4.3.2.1 User interface 213
4.3.2.2 Database query and maintenance 215
4.3.2.3 The robot crawling algorithm 218
4.3.2.4 The robot parsing/database entry algorithm 220
4.3.3 Design and implementation constraints *224*
4.4 CHAPTER CONCLUSION 226

CHAPTER FIVE 227

DISCUSSION AND EVALUATION 227

5.1 METADATA PARSING AND DATABASE ENTRY 228
5.2 ON-LINE METADATA EDITING UTILITY 235
5.3 GEOSPATIAL DATA-CONVERSION UTILITIES 237
5.4 INTEGRATED WWW SEARCH-ENGINE 247
5.5 GEOSPATIAL WEB ROBOT 250
5.6 CHAPTER CONCLUSIONS 252

CHAPTER SIX **254**

CONCLUSIONS AND FUTURE RESEARCH **254**

6.1 AN ON-LINE FEDERATED DATABASE FOR GIS DATA 256

6.2 AN ON-LINE DECISION SUPPORT SYSTEM FOR GIS DATA 264

6.3 THE FUTURE OF GIS DATA ON THE WEB 270

6.4 FINAL REMARKS 276

REFERENCES **278**

APPENDIX A **299**

THE GMS COMPACT DISK 299

APPENDIX B **301**

INTERCHANGE STANDARDS 301

APPENDIX C **307**

ATTRIBUTE ASSIGNMENT FOR THE E-R MODEL 307

APPENDIX D **312**

THE MSOFT UTILITY 312

APPENDIX E **316**

A CSDGM METADATA FILE AND MPARSER OUTPUT 316

APPENDIX F **327**

THE HTTP AND FTP UPLOAD UTILITY 327

APPENDIX G **329**

THE GEOMETRIC SEARCH SUBROUTINES 329

LIST OF FIGURES

FIGURE 1.1 THE CORE OF GMS IS THE METADATA DATABASE COMPONENT. IT IS THE LOCAL REPOSITORY FOR METADATA INFORMATION COLLECTED AND STORED FROM THE WEB. THERE ARE FIVE PROCESSING COMPONENTS LINKED TO THE DATABASE. THE EDITING AND PARSING FUNCTIONS OF GMS ARE COLLECTIVELY CALLED THE METADATA ENTRY/MANIPULATION COMPONENTS OF THE SYSTEM. 14

FIGURE 2.1 THE DIGIMAP IMPLEMENTATION SHOWING THE RDBMS METADATABASE AND ITS RELATION TO OTHER SYSTEM COMPONENTS, IN PARTICULAR THE HTTP SERVER (FROM MEDYCKYJ-SCOTT AND MORRIS, 1998) 36

FIGURE 2.2 SOME OF THE COMPONENTS ASSOCIATED WITH A GENIE NODE INCLUDING THE MASTER DIRECTORY DESIGNED TO REFERENCE LOCAL AND REMOTE METADATA FILES (FROM GENIE PROJECT, 1995) 38

FIGURE 2.3 DUAL SERVER CONFIGURATION FOR THE NATIONAL GEOSPATIAL DATA CLEARINGHOUSE (FROM NEBERT, 1995B). WEB CLIENTS CAN ACCESS THE SEARCH-ENGINE USING THE HTTP SERVER VIA CGI SCRIPTS, WHICH, IN TURN, ACCESSES THE Z39.50 SERVER AS A BACKEND APPLICATION. Z39.50 CLIENTS CAN ACCESS THE SEARCH-ENGINE DIRECTLY USING A SERVER THAT IMPLEMENTS THE Z39.50 PROTOCOL. 40

FIGURE 2.4 A SMALL PORTION OF THE WEB INTERFACE FOR THE BIC METADATA FORM SHOWING THE COLOUR CODE SCHEME USED TO LABEL METADATA ELEMENTS AS BEING MANDATORY, MANDATORY IF APPLICABLE OR OPTIONAL. THE DESCRIPTION OF METADATA ELEMENTS AND HTML FORM ELEMENTS ARE A COMMON FEATURE OF SUCH METADATA ENTRY UTILITIES 51

FIGURE 2.5 A PORTION OF THE *IDENTIFICATION INFORMATION* FORM FOR THE METAMORPH ON-LINE METADATA ENTRY UTILITY. IT SHOWS THE FORM-RETRIEVAL SERVICE THAT CAN RETRIEVE FORMS LOCATED ON A LOCAL SERVER OR REMOTELY VIA A URL. THE LOWER SECTION CONSISTS OF TWO FREE-FORM TEXT FIELDS IN WHICH TO ENTER, IN THIS CASE, SUMMARY AND PURPOSE INFORMATION. 53

FIGURE 2.6 THE DATA REGISTRATION TOOL THAT CREATES A CUSTOMISED HTML FORM FOR ENTERING DIF METADATA TO ONE OR MORE GCMD SERVERS. THE SYSTEM ALLOWS DATA-PROVIDERS TO CONSTRUCT A FORM FOR ONLY AVAILABLE METADATA INFORMATION. SELECTING ALL THE CHECKBOXES ABOVE PRODUCES THE ENTIRE FORM FOR THE METADATA RECORD CONTAINING ALL OF THE FIELDS SPECIFIED IN THE DIF STANDARD. 56

FIGURE 2.7 A PORTION OF A TABLE PROVIDING TOOLS THAT ALLOW MODIFICATION OF DATA SET RECORDS (DIFS). THE FAR-LEFT COLUMN SPECIFIES TOOLS FOR MODIFYING DIFS THAT RESIDE EITHER ON THE GCMD DATABASE, THE DATA-PROVIDER'S DISK SPACE OR ANYWHERE ELSE ON THE WEB. THE FAR-RIGHT COLUMN PROVIDES LINKS TO THESE TOOLS..... 57

FIGURE 2.8 A CONNECTION-ORIENTED DATA TRANSMISSION SERVICE USING THE CLIENT/SERVER MODEL OPERATING OVER DEPARTMENTAL LOCAL AREA NETWORK (LAN), CORPORATE WIDE AREA NETWORK (WAN) OR INTRANETS (SEE PRETTY, 1992 FOR A DISCUSSION OF THESE TYPES OF NETWORKS). 63

FIGURE 2.9 A 'CONNECTIONLESS' AND 'STATELESS' DATA TRANSMISSION SERVICE USING THE REQUEST/RESPONSE MODEL OPERATING OVER THE INTERNET. 64

FIGURE 2.11 THE SCOTLIS USER-INTERFACE DISPLAYING THE APPLICATIONS MENU IN A SEPARATE FRAME TO THE LEFT AND THE MAIN DISPLAY AREA TO THE RIGHT. 72

FIGURE 2.12 THE GCMD GUIDED SEARCH-INTERFACE THAT ASSEMBLES DATABASE QUERIES WHICH ACCESS THE ON-LINE MASTER DIRECTORY. EACH BUTTON TO THE LEFT REVEALS ADDITIONAL FORM COMPONENTS DESIGNED TO NARROW THE SEARCH QUERY FOR A PARTICULAR CRITERION..... 75

FIGURE 3.1 ENTITY-RELATIONSHIP MODEL OF THE GMS METADATABASE. THE PRIMARY_KEY FOR THE IDENTIFICATION_ INFORMATION TABLE IS THE COLUMN THAT IDENTIFIES A DATABASE RECORD. ALL OTHER TABLES ARE RELATED TO THE I-I TABLE BY EITHER ONE-TO-ONE, ONE-TO-MANY OR MANY-TO-MANY RELATIONSHIPS. APPENDIX C OUTLINES THE ATTRIBUTE ASSIGNMENTS FOR EACH TABLE. 83

FIGURE 3.2 MPARSER AND MLOADER WITHIN GMS. 103

FIGURE 3.3 FLOW DIAGRAM FOR THE DATABASE ENTRY AND SOFTWARE ACCESS UTILITY. THE DIAGRAM TO THE LEFT EXECUTES THE MSOFT UTILITY (APPENDIX D) WHILE THE RIGHT EXECUTES THE MPARSER AND MLOADER UTILITIES ONE AFTER ANOTHER. 106

FIGURE 3.4 FLOW DIAGRAM FOR THE MPARSER UTILITY. 107

FIGURE 3.5 THE MASTER PROGRAM FOR THE MLOADER DATABASE ENTRY UTILITY. EACH METADATABASE TABLE IS LABELLED WITH AN ACRONYM (E.G. II FOR *IDENTIFICATION_ INFORMATION*). THE OPERATION OF EACH PARSING PROGRAM IS GENERALISED BY THE DIAGRAM OF FIGURE 3.6. THE OPERATION OF EACH DATABASE ENTRY PROGRAM IS GENERALISED BY THE DIAGRAMS OF FIGURE 3.7. 108

FIGURE 3.6 THE PARSING PROGRAM CONSTRUCTED FOR EACH METADATABASE TABLE. 108

FIGURE 3.7 THE DATABASE ENTRY PROGRAM CONSTRUCTED FOR EACH METADATABASE TABLE. THE ABOVE BRANCH IS RUN IF THE ROW CONTAINS NO METADATA VALUES. THIS IS DONE TO ENSURE PRIMARY AND FOREIGN KEYS ARE INCREMENTED, WHICH MAINTAINS THE REFERENTIAL INTEGRITY OF THE DATABASE. 109

FIGURE 3.8 THE USER-INTERFACE FOR THE METADATA PARSING AND DATABASE ENTRY UTILITY. USER CAN UPLOAD A TEXT FILE CONTAINING METADATA AND PROCESS IT USING THE MSOFT OR MLOADER UTILITIES VIA THE RADIO BUTTONS, RESPECTIVELY. 110

FIGURE 3.9 A PORTION OF THE FIELD.TXT FILE SHOWING THE METADATA ELEMENTS FOR THE *IDENTIFICATION_INFORMATION* TABLE. 112

FIGURE 3.10 MPARSER PROGRAM ITERATIONS FOR THE SECTION.TXT AND FIELD.TXT FILE. 116

FIGURE 3.11 EXTRACTING METADATA FROM A METADATA FILE AND ITS SUBSEQUENT DATABASE ENTRY 118

FIGURE 3.12 THE PARSING ROUTINE FOR THE MLOADER PARSING PROGRAMS. THE FILE IS PROCESSED USING A SERIES OF EMBEDDED STRUCTURES. EACH LEVEL OF EMBEDDING IS USED TO ITERATE AROUND ONE LEVEL OF MULTIPLICITY ENCOUNTERED IN THE FILE. 120

FIGURE 3.13 THE PROGRAM EXECUTION SEQUENCE OF THE MLOADER DATABASE ENTRY PROGRAM FOR A SINGLE METADATA RECORD. THE FIRST LOOP EXECUTES PROGRAMS THAT ENTER METADATA FROM THE LOAD-FILE TO THE DATABASE WITH 'OCCURRENCE' VALUES OF 1. THE SECOND, THIRD AND FOURTH LOOPS HAVE 'OCCURRENCE' VALUES OF 2, 3 AND 4, RESPECTIVELY. 123

FIGURE 3.14 THE ON-LINE METADATA EDITING UTILITY. EACH LINK CONNECTS TO AN HTML FORM PROVIDING DATABASE ENTRY AND MODIFICATION FACILITIES FOR THE NAMED TABLE. NEW METADATA RECORDS ARE CREATED USING THE IDENTIFICATION_ INFORMATION FORM. 131

FIGURE 3.15 THE FLOW DIAGRAM FOR THE CGI SCRIPT USED TO PRE-PROCESS METADATA INFORMATION SUPPLIED BY THE DATA-PROVIDER VIA AN HTML FORM AND SUBSEQUENTLY ENTERED INTO THE *IDENTIFICATION_INFORMATION* TABLE OF THE METADATABASE. 132

FIGURE 3.16 THE FLOW DIAGRAM FOR THE CGI SCRIPT USED TO PRE-PROCESS METADATA INFORMATION SUPPLIED BY THE DATA-PROVIDER VIA THE HTML

FORMS AND SUBSEQUENTLY ENTERED INTO THE VARIOUS TABLES (EXCLUDING *IDENTIFICATION_INFORMATION*) OF THE METADATABASE. 132

FIGURE 3.17 A GENERALISED DESIGN OF ALL THE DATABASE ENTRY/MODIFICATION PROGRAMS. THERE IS ONE PROGRAM FOR EACH METADATA DATABASE TABLE. EACH PROGRAM IS CAPABLE OF INSERTING (DIAGRAM A), UPDATING (DIAGRAM B) AND SELECTING (DIAGRAM C) METADATA VALUES IN TO AND FROM DATABASE TABLES. 134

FIGURE 3.18 THE IDENTIFICATION_INFORMATION METADATA ENTRY FORM SHOWING THE ROW NUMBER, USERNAME, PASSWORD, SOME FIELDS AND THE SUBMIT BUTTON TOWARDS THE BOTTOM. THE ENTIRE FORM IS FAR TOO LONG TO BE DISPLAYED ON THIS PAGE; THEREFORE, ONLY THE TOP AND BOTTOM PORTIONS ARE SHOWN. 136

FIGURE 3.19 THE ROW NUMBER, USERNAME AND PASSWORD VERIFICATION FOR THE CITATION_ INFORMATION METADATA ENTRY FORM. 138

FIGURE 3.20 THE ROW NUMBER FIELDS USED TO SPECIFY SOURCE RELATED CITATION INFORMATION. 140

FIGURE 3.21 THE HTML FORM BUTTON USED TO CONFIRM THE CREATION OF A NEW METADATABASE RECORD. 142

FIGURE 3.22 METADATA VALUES (ROW-NUMBER, FULLNAME, PASSWORD, ETC.) CAN BE INSERTED OR SELECTED TO AND FROM THE DATABASE. IF IT IS THE LATTER, THE RETRIEVED METADATA CAN BE UPDATED OR DELETED FROM ANOTHER HTML OR CREATED ON-THE-FLY. 150

FIGURE 4.1 THE CGI SCRIPT PROCESSES USER-SPECIFIED VALUES FROM THE INTERFACE BEFORE SENDING THEM TO THE GMS SEARCH-ENGINE. THIS IS ACCOMPLISHED WITHIN THREE SEPARATE BRANCHES OF THE PROGRAM THAT DEFINE DATA, LOCATION AND KEYWORD SEARCH CRITERIA. 157

FIGURE 4.2 THE GMS SEARCH-ENGINE RECEIVES THE VALUES FROM THE CGI SCRIPT AND QUERIES THE DATABASE DEPENDING ON THE SEARCH CRITERIA. IT PRINTS TO THE BROWSER, METADATA AND HTML EXECUTABLE BUTTONS USED TO RUN ON-LINE SOFTWARE UTILITIES. 158

FIGURE 4.3 THE USER-INTERFACE USED TO QUERY THE METADATA DATABASE. QUERIES CAN BE BY DATE, LOCATION OR KEYWORD SEARCH CRITERION. SEARCH-AREAS CAN CONSIST OF CIRCLES, RECTANGLES OR QUADRILATERALS. 160

FIGURE 4.4 A GEOGRAPHICAL CO-ORDINATE IN DEGREES MINUTES AND SECONDS. .. 164

FIGURE 4.5 SOME EXAMPLES OF CIRCLE AND QUADRILATERAL SEARCH PATTERNS... 167

FIGURE 4.6 CALCULATION FOR LINE-LINE INTERSECTION, CIRCLE-LINE INTERSECTION AND LENGTH BETWEEN TWO POINTS (POINT-DISTANCE) FOR THE CIRCLE AND QUADRILATERAL SEARCH PATTERS..... 168

FIGURE 4.7 TESTING SEQUENCE FOR THE CIRCLE SEARCH PATTERN. SEE THE TEXT FOR A DETAILED EXPLANATION OF EACH TEST..... 169

FIGURE 4.8 TESTING SEQUENCE FOR THE QUADRILATERAL SEARCH PATTERN. SEE THE TEXT FOR A DETAILED EXPLANATION OF EACH TEST..... 170

FIGURE 4.9 THE MAIN DATABASE QUERY FOR THE GMS SEARCH-ENGINE WHICH RETURNS ALL THE DISTINCT ROWS FOUND FROM EACH OF THE SIX DATABASE QUERIES..... 171

FIGURE 4.10 THE DIGITAL TRANSFER INFORMATION TABLE IS REPRODUCED FROM FIGURE 3.1. IT SHOWS, IN PARTICULAR, THE FORMAT_NAME AND NETWORK_ADDRESS METADATA ELEMENTS WHOSE VALUES ARE DRAWN FROM THE DATABASE AND SUPPLIED TO ON-LINE SOFTWARE UTILITIES..... 172

FIGURE 4.11 A DATABASE QUERY OF A METADATA RECORD USING THE GMS SEARCH-ENGINE..... 173

FIGURE 4.12 THE BROWSE_GRAPHIC TABLE FROM THE GMS METADATABASE..... 174

FIGURE 4.13 AN HTML EXECUTABLE 'BUTTON' TO BROWSE ON-LINE GRAPHIC FILES. 174

FIGURE 4.14 SAMPLE CODE FROM THE METADATABASE SEARCH-ENGINE..... 175

FIGURE 4.15 ARC/INFO DATA-CONVERSION COMMANDS USED IN GMS..... 180

FIGURE 4.16 FLOW DIAGRAM FOR THE DXFARC, DLGARC AND TIGERARC GEOSPATIAL DATA-CONVERSION UTILITIES. THE LEFT SIDE OF THE DIAGRAM SHOWS HOW FILES ARE UPLOADED VIA HTTP; THE RIGHT SIDE USES FTP TO UPLOAD FILES. THERE IS A DATA-CACHE ATTACHED TO EACH OF THE UPLOADING PROGRAMS..... 182

FIGURE 4.17 FLOW DIAGRAM FOR THE AML USED IN THE DLGARC GEOSPATIAL DATA-CONVERSION UTILITY. THE CONVERSION COMMAND PROVIDES THE OPTION TO CREATE A POINT COVERAGE FROM THE DEGENERATE LINE RECORD IN THE DLG DATA..... 183

FIGURE 4.18 FLOW DIAGRAM FOR THE AML USED IN THE TIGERARC GEOSPATIAL DATA-CONVERSION UTILITY. THE CONVERSION COMMAND PROVIDES THE OPTION TO CREATE COVERAGES CONTAINING EITHER POINT FEATURES, LANDMARK FEATURES OR BOTH. THE UTILITY PERFORMS ALSO OPTIONAL COVERAGE PROJECTION AND TOPOLOGY CREATION..... 184

FIGURE 4.19 FLOW DIAGRAM FOR THE AML USED IN THE DXFARC GEOSPATIAL DATA-CONVERSION UTILITY. THE CONVERSION COMMAND MUST AUTOMATICALLY SET-UP A 'WATCH' FILE THAT CONTAINS INFORMATION FROM A DXFINFO COMMAND. THE FILE IS EXAMINED FOR DXF LAYERS AND AN ARC/INFO COVERAGE IS CREATED FOR EACH LAYER. 184

FIGURE 4.20 FLOW DIAGRAM FOR THE ARCDXF GEOSPATIAL DATA-CONVERSION UTILITY. THE UTILITY UPLOADS TO GMS SEPARATELY ONE OR MORE ARC, POINT AND ANNOTATION COVERAGES. THEY ARE USED TO CREATE A DXF DATA SET CONTAINING ONE OR MORE DXF LAYERS. 185

FIGURE 4.21 FLOW DIAGRAM FOR THE ARCDLG GEOSPATIAL DATA-CONVERSION UTILITY. THE UTILITY UPLOADS A COVERAGE, AN OPTIONAL POINT COVERAGE, A FILE CONTAINING PROJECTION PARAMETERS AND A DLG HEADER FILE. 186

FIGURE 4.22 FLOW DIAGRAM FOR THE ARCTIGER GEOSPATIAL DATA-CONVERSION UTILITY. THE CONVERSION COMMAND ACCEPTS A COVERAGE CONTAINING LINE, POINT AND/OR LANDMARK FEATURES. THEY ARE IMPORTED AND SUBSEQUENTLY CONVERTED TO A TIGER/LINE DATA SETS. 187

FIGURE 4.23 THE E00 FILE DECOMPRESSION UTILITY. THE UTILITY CONVERTS COVERAGES TO STANDARD ASCII FORMAT OVER THE WEB FOR USE IN OTHER TYPES OF GIS SOFTWARE. 188

FIGURE 4.24 THE WEB BASED ARCPLOT UTILITY. THE UTILITY UPLOADS A COVERAGE IN E00 FORMAT, CONVERTS IT TO A CGM, GRA, EPS OR ADOBE PLOT FILE AND RETURNS IT TO THE CLIENT 189

FIGURE 4.25 HTTP AND FTP FILES UPLOAD FACILITY USED IN ALL THE ON-LINE GMS UTILITIES. 190

FIGURE 4.26 THE INTERFACE COMPONENTS OF THE DXFARC, DLGARC AND TIGERARC CONVERSION UTILITIES. 191

FIGURE 4.27 A PORTION OF THE USER-INTERFACE FOR THE E00 DECOMPRESSION UTILITY THAT SHOWS THE DIFFERENT COMPRESSION PROGRAMS AVAILABLE TO COMPRESS THE RESULTING E00 ASCII FILE. 193

FIGURE 4.28 A PORTION OF THE USER-INTERFACE FOR THE ARCPLOT UTILITY SHOWING THE PLOT PARAMETERS THAT INCLUDE THE FEATURE TYPE, SYMBOL TYPE AND OUTPUT FILE FORMAT. 193

FIGURE 4.29 A PORTION OF THE DLGARC USER-INTERFACE SHOWING TWO FIELDS THAT REQUIRE THE NAMES OF THE CONVERTED COVERAGE AND THE OPTIONAL POINT COVERAGE. 198

FIGURE 4.30 A PORTION OF THE TIGERARC USER-INTERFACE SHOWING THE OPTIONAL COVERAGE FIELDS FOR THE POINT AND LANDMARK FEATURES. 198

FIGURE 4.31 THE STRUCTURE OF AN HTML EXECUTABLE BUTTON. THE TOP SERIES OF BOXES IS A HYPERTEXT LINK TO THE DATA SETS. THE EXECUTABLE BUTTON IS A SIMPLE HTML FORM WITH ONLY TWO TYPES OF FORM ELEMENTS, A SUBMIT-BUTTON AND VARIOUS HIDDEN FIELDS. THE HIDDEN FIELDS CONTAIN COMMAND-LINE ARGUMENTS, WHICH ARE SENT TO THE APPLICATION REFERENCED IN THE SECOND LINE OF THE ABOVE FIGURE. 200

FIGURE 4.32 DATA-SWITCHYARD ARCHITECTURE (FROM PRADHAN AND GITTINGS, 1996). 201

FIGURE 4.33 A SET OF BOXES REPRESENTING THE AML COMMANDS USED IN THE ARCDXF, ARCDLG AND ARCTIGER CONVERSION UTILITIES. THE TOP SET OF BOXES DETERMINES THE COVERAGES TO BE IMPORTED AND THE MIDDLE BOX REPRESENTS THE DATA CONVERSION. THE BOTTOM SET OF BOXES DETERMINES THE COVERAGES TO BE DELETED. 204

FIGURE 4.34 THE UPLOAD FACILITY FOR THE ARCPLOT UTILITY DESIGNED TO UPLOAD THE DATA SET ONCE AND ACCESS IT CONTENTS ON THE LOCAL GMS SERVER THEREAFTER. 206

FIGURE 4.35 GEOBOT INTERFACE ARCHITECTURE. 209

FIGURE 4.36 FLOW CHART OF DATABASE QUERY ALGORITHM. THIS SMALL SEARCH-ENGINE IS CAPABLE OF BOOLEAN 'AND/OR' TYPE QUERIES. 210

FIGURE 4.37 FLOW CHART OF URL DELETING ALGORITHM CAPABLE OF REMOVING A SINGLE URL FROM THE DATA FROM THE DATABASE OR AN ENTIRE SET OF URLS THAT SHARE THE SAME DOMAIN (I.E. WWW.GEO.ED.AC.UK) NAME. 210

FIGURE 4.38 FLOW CHART OF GEOBOT'S 'CRAWL' ALGORITHM THAT ALLOWS THE WEB ROBOT TO MOVE FROM ONE HTML PAGE TO ANOTHER. 211

FIGURE 4.39 FLOW CHART OF THE GEOBOT PARSING ALGORITHM. NOTE THE ROBOT'S ABILITY TO IDENTIFY HTML FRAME PAGES, RELATIVE URLS, HTTP/FPT URLS AND OTHER INTERNET PROTOCOLS. 212

FIGURE 4.40 THE GEOBOT DATABASE QUERY PAGE. IT CONSISTS OF A SINGLE FIELD WHERE ONE OR MORE SEARCH-TERMS OR KEYWORDS CAN BE ENTERED AND JOINED WITH BOOLEAN OPERATORS. 213

FIGURE 4.41 THE 'DELETE' PAGE. HYPERTEXT LINKS CAN BE DELETED FROM THE DATABASE (EITHER A SINGLE PAGE OR AN ENTIRE SITE DOMAIN) GIVEN A URL ADDRESS AND VERIFICATION INFORMATION. 214

FIGURE 4.42 THE ROBOT 'SUBMIT' PAGE. A HYPERTEXT LINK CAN BE ENTERED WHICH SPECIFIES A STARTING HTML PAGE USED BY THE ROBOT TO LAUNCH ITS SEARCH. BY SPECIFYING A FTP SITE THE ROBOT IS DIRECTED TO SEARCH FOR ALL FTP LINKS OF THAT PARTICULAR DOMAIN. 215

FIGURE 4.43 GEOBOT USES THE CONTENTS WITHIN THE <TITLE> TAG (IN BOLD) AS A BRIEF DESCRIPTION OF THE FOUR LINKS PARSED IN THE ABOVE PAGE AS IT STORES THE INFORMATION WITHIN ITS DATABASE. 216

FIGURE 4.44 MECHANISM THE ROBOT USES TO ACQUIRE NEW URL ADDRESSES TO TRAVERSE A SITE DOMAIN. 219

FIGURE 4.45 THE 'SIEVE' LIKE PARSING MECHANISM FOR GEOBOT. HYPERTEXT LINKS ARE INDIVIDUALLY EXAMINED BY PLACING THE RELEVANT ONES IN SEPARATE C LANGUAGE ARRAYS FOR FURTHER USE LATER IN THE PROGRAM. 222

FIGURE 5.1 RADIO BUTTONS AND HTML TEXT FIELDS WERE ADDED TO FACILITATE THE ACCESS OF ON-LINE FILES LOCATED ON FTP SERVERS. EACH RADIO BUTTON PROVIDES A DIFFERENT FORMAT FOR THE 'GET FILE' COMMAND. THE TEXT FIELDS ALLOW USERS TO ENTER LOGIN AND PASSWORD INFORMATION TO ACCESS FILES FROM THEIR OWN SERVERS. 241

FIGURE 6.1 A DIAGRAM OF THE PROPOSED FEDERATED METADATA DATABASE WHERE USERS ARE ABLE TO QUERY MULTIPLE SERVERS CONTAINING STANDARDISED METADATA AND ACCESS GEOSPATIAL DATA REFERENCED THROUGH HYPERLINKS FOR CONVERSION TO ONE OR MORE DIFFERENT DATA FORMATS 260

LIST OF TABLES

TABLE 2.1 SUMMARY OF SOME METADATA EDITORS.....48

TABLE 3.1 INSERTING KEYWORDS INTO THE KEYWORD TABLES.....145

TABLE 3.2 UPDATING KEYWORDS FROM THE KEYWORD TABLES.....145

TABLE 4.1 THE DATE DATATYPE.....162

TABLE 4.2 SEARCH ENGINE DEMONSTRATION.....217

TABLE 4.3 THE GEOBOT DATABASE.....221

LIST OF ACRONYMS

AACR2 - Anglo-American Cataloguing Rules

AI - Artificial Intelligence

AML - Arc Macro Language

AMS - Automated Mapping System

ANSI - American National Standards Institute

API - Applications Programming Interface

ARCE - ARC/INFO Export format

BLMCSO - U.S. Bureau of Land Management, Colorado State Office

CDF - Common Data Format

CEOS IDN - Committee on Earth Observation Satellites International Directory Network

CGI - Common Gateway Interface

CLI - Call Level Interface

CORBA - Common Object Request Broker Architecture

CSDGM - Content Standard for Digital Geospatial Metadata

CSS - Cascading Style Sheets

DCL - Digital Command Language

DCOM - Distributed Common Object Model

DGI - Distributed Geographic Information

DHTML - Dynamic HTML

DIF - Directory Interchange Format

DIME - Dual Independent Map Encoding

DLG - Digital Line Graph

DOM - Document Object Model

DSA - Directory System Agent

DSS - Decision Support Systems

DTD - Document Type Definition

DXF - Auto CAD Drawing Interchange File

ESRI - Environmental Systems Research Institute

FGDC - Federal Geographic Data Committee

FTP - File Transfer Protocol

GCMD - Global Change Master Directory

GEC - Global Environmental Change

GEMIP - Global Earthquake Mapping and Information Package

GIS - Geographic Information System

GMS - Geospatial Metadata Server

GUI - Graphical User Interface

HTML - HyperText Markup Language

HTTP – HyperText Transfer Protocol

IGDS - Interactive Graphic Design Software

IGES - Initial Graphics Exchange Standard

IIOP - Internet Inter-ORB Protocol

IP - Internet Protocol

ISO - International Standards Organisation

JANET - Joint Academic Network

JDBC - Java Database Connection

LAN - Local Area Network

MIADS - Map Information Assembly Display

NASA - National Aeronautics and Space Administration

NBII - National Biological Information Infrastructure

NEIC - USGS National Earthquake Information Centre

NGDC - National Geospatial Data Clearinghouse

NGDF - National Geospatial Data Framework

NISO - National Information Standards Organisation

NSDI - National Spatial Data Infrastructure

NTF - Neutral Transfer Format

ODBC - Open Database Connectivity

OGC - Open GIS Consortium

OLAP – On-line Analytical Processing

OLE - Object Linking and Embedding

ORB - Object Request Broker

OS - Ordnance Survey

OSI - Open Systems Interconnections

PIA - Personal Information Agents

RDBMS - Relational DataBase Management System

RPC - Remote Procedure Call

SAIF - Spatial Archive and Interchange Format

SDE – Spatial Database Engine

SDTS - Spatial Data Transfer Standard

S-HTTP - Secure HTTP

SQL - Sequential Query Language

SSL - Secure Socket Layer

TCP - Transmission Control Protocol

TCP/IP - Transmission Control Protocol/Internet Protocol

TIGER - Topologically Integrated Geographic Encoding and Referencing Systems

URL - Uniform Resource Locator

USGS - United States Geological Survey

W3C - WWW Consortium

WAIS - Wide Area Information Service

WAN - Wide Area Network

WWW - World Wide Web

XML - Extensible Mark-up Language

CHAPTER ONE

INTRODUCTION

1.1 The Aim

The aim of this thesis is to understand fundamental issues and software concepts that influence the storing, cataloguing and processing of geographical data over the World Wide Web (WWW or Web). This is achieved by in part developing and evaluating a software system designed to operate in such a computing environment.

The Internet is as a powerful technology with which to transfer large amounts of information. The combination of GIS[1] and Web technologies provide an essential service to a growing community of GIS users. The technology enables Web users to download geographical data[2] easily by searching multiple on-line data storage facilities. At the same time it allows providers the means to conveniently distribute their data to a vast number of GIS users. The expenditures incurred in the production of geospatial data[3] as part of a GIS project provide an important incentive for data-users to explore new techniques in sharing data amongst themselves (Waker et al., 1992). For dedicated data-providers this research gives insight into the technical and organisational issues that

[1] A GIS or Geographic Information System, used in both a singular and plural sense throughout this dissertation, is a computer-based system or systems designed to collect, store and analyse objects and phenomena where geographic location is an important characteristic or critical to the analysis (Aronoff, 1989).

[2] Data or geospatial data is used in a singular or plural sense throughout this thesis.

[3] Geospatial data is the information used within a GIS that identifies the geographic location and characteristics of natural or constructed features and boundaries on the earth and is designed according to a pre-specified format for operation within software that supports such a format.

must be considered before using such software systems to distribute geospatial data over the Web.

1.2 Rationale

Only within this decade have distributed networks begun transferring large volumes of vector and image data. Indeed, the trend towards the utilisation of geographic information is growing exponentially (Newton et al., 1992a). Many organisations produce large amounts of data from various projects over the course of time but this information is frequently poorly distributed to a wider audience. The results of such projects are often distributed through textual reports that are rarely seen outside the organisation. The data collected from these projects is usually never included in these reports. The only option for users requiring access to such data resources is through direct contact with someone within the organisation and associated with the particular project. By contrast, sharing geospatial data is of considerable benefit to small organisations and academic institutions where resources are scarce and the costs of data from commercial or government suppliers are substantial.

A further problem exists in that primary data collection is expensive. Avoiding this expense by using pre-existing data is possible but locating suitable data also proves difficult. One of the primary expenditures incurred in GIS implementation is from data collection and base mapping (Dickinson and Calkins, 1988; Rhind, 1992; Douglas and Tomlinson, 1992). The data required by an organisation may be created already by other individuals or organisations; however, knowledge of its existence is often lacking and the means to access the data in a timely fashion is, at present, somewhat deficient. Most organisations that utilise the capabilities of GIS must spend considerable resources in developing systems. Such investment usually demands a minimum return or, at least, a perceived return in order for management to realise any additional project funding.

Therefore, the ability for such organisations to acquire pre-existing data substantially reduces their overall project costs and facilitates project completion.

The WWW offers a solution to some of these problems of data-sharing across distributed systems (Calkins, 1992). It allows users and providers to search, access and distribute geospatial data using a variety of effective mechanisms, many of which are still being developed and tested. The new technology also provides a culture for geospatial data sharing where resources are used effectively by all those concerned and costs are minimised. By using the Web, users of geospatial data are able to modify the geospatial data acquired from data-providers and redistribute it to the wider GIS community thereby becoming data-providers themselves. A similar analogy in the software industry is the concept of freeware/shareware where software is distributed either for free or at a nominal cost, perhaps modified or enhanced and then passed on to other software developers or the wider market as a whole. A number of Web-based technologies and protocols are needed to help implement the ease of data-exchange over the Web. Primary amongst these are standardisation of the geospatial data formats themselves and also of the metadata that describes them. In addition, utilities that assist the accessing and searching of remote databases and those which permit conversion between geospatial data formats are key requirement for practical usability.

As context to this general background, certain key questions are identified. These provide insight into the issues that must be addressed in designing a software system capable of facilitating the on-line exchange of geospatial data. Newton et al. (1992b) as well as Goodchild, 1995 provides additional insight into the issues concerning the transfer of spatial information over distributed networks. There are a number of different metadata standards. Metadata is data about the content, quality, condition, and other characteristics of geospatial data and is used to provide documentation for such data products. Therefore, the following questions are put forward with regard to how such metadata is represented on the Web, how it is stored and how it is maintained:

What metadata standard do developers of geospatial data dissemination sites use to describe geospatial data and does everyone use the same metadata standard?

Because different standards' bodies developed these standards, the terminology used to describe geospatial data tends to radically differ. This lack of common standardisation, particularly with regard to geospatial data dissemination Web sites, precludes the user from knowing the format of the metadata and how to interpret it. Many Web sites use ad-hoc database schemes to store metadata, which also prevent them from integrating their indexing and query capabilities with those of other HTTP[4] servers because each use locally-developed database schemes that are incompatible with each other. There are currently only a few Web-based examples of complete or even partial on-line implementations of the CSDGM[5] standard (Mapping Science Committee, 1993).

Standardised metadata helps publicise and support the geospatial data produced by organisations to an increasingly wider GIS user community. Many data dissemination systems operating on the Internet rely on the predictability of form and content that a metadata standard provides. Predictability is assured only by conformance to standards. Not only is this metadata standard expressive of the geospatial data it is required to describe but it ensures that data-providers and data-users both speak the same language and arrive at the same understanding when accessing on-line geospatial data (Schweitzer, 1996a).

How is the metadata stored for on-line access, within text files, DBMS or a combination of the two?

---

[4] A HyperText Transfer Protocol (HTTP or Web) server is a class of software designed to use a standardised protocol to process requests for information from clients over a computer network and initiate an appropriate response. This is a process known the request/response model for client/server communication. See chapter 2, section 2.5.2 for further information.

[5] Content Standard for Digital Geospatial Metadata (CSDGM) is one of a number of metadata standards that specify the information content of metadata for a set of digital geospatial data by providing a common set of terminology and definitions (Federal Geographic Data Committee, 1994). See section 2.1.1 for a fuller explanation.

Using a DBMS[6] to store metadata substantially increases the flexibility of search and indexing operations that could not otherwise be achieved using less sophisticated non-proprietary means of information storage. A DBMS provides powerful facilities for storage, query and report of metadata and allows the on-line data-provider the freedom to determine:

- How the metadata is searched and by which sections or elements.
- How much information is provided to the user.
- How this information is structured and formatted.
- How the information is viewed.

At present, on-line metadata usually exists as standardised text or HTML[7] files, or is contained in a DBMS using an ad-hoc database scheme that provides only a few descriptive fields. These comprise a bare minimum of information required to adequately describe a geospatial data set. Limited ad-hoc database implementations possess table and column names that tend to vary from site to site forcing the development of customised search-engines for each Web server. Querying entire metadata text or HTML files creates its own problems because the excessive length of most metadata files tends to initially confront the user with too much information. Sifting through these files to locate required information entails the examination of specific sections and elements before deciding on the appropriateness of a particular geospatial data set. It is possible to parse information from the metadata files but this requires considerable time and processing which are two commodities that are at a premium on the Internet. Also, combining the use of flat files and a DBMS excludes the rapid query of other types of metadata information and tends to place too much control with the site administrator in determining what metadata elements should or should not be queried on-line.

[6] A DataBase Management System is a class of software that consists of a collection of algorithms that performs operations, such as the storage, retrieval and modification on a centralised repository of information, organised using a predetermined pattern or database schema.

What is an effective method for data-providers and administrators of geospatial data dissemination sites to enter an entire metadata file into an on-line relational database?

It is impractical to manually enter such large amounts of metadata into a database because of the sheer size of metadata records. Therefore, the ability to parse metadata files and load the information directly into the columns and tables of an on-line DBMS is of considerable benefit to providers of geospatial data and administrators of data dissemination sites. For example, a CSDGM compliant metadata file possesses a maximum of 334 different elements and 119 compound elements (Schweitzer, 1996a). However, the use of automatic parsing of metadata files and entry into on-line databases provides a means to efficiently maintain the currency of metadata databases. The process allows site administrators to screen metadata files that are submitted via the Web before entering the documents into their on-line metadata catalogues.

Automatic metadata parsing and database entry also allows site administrators to relegate much of the responsibility for database maintenance to the geospatial data-providers. In such a situation, a group or community of geospatial data-providers located anywhere on the Web are able to quickly build a comprehensive centralised metadata catalogue of their geospatial data sets. In fact, such a system can be used to submit metadata to a number of remote on-line metadata catalogues, which are accessed by users of geospatial data from anywhere on the Internet.

What is an effective method for data-providers to remotely enter and update individual metadata elements into an on-line relational database?

Ideally, geospatial data-providers are able to submit formally-structured and compliant metadata to on-line geospatial data cataloguing sites in a way that divides the arduous task of forms-based textual entry into several separate parts. The amount of information contained within a metadata file that is necessary to represent appropriately one or more

[7] HyperText Markup Language (HTML) is a mark-up language used to prepare WWW documents (Graham, 1996).

geospatial data sets is well beyond what the producer of the geospatial data is willing to enter into an on-line metadata catalogue in a single sitting. Facilities for the on-line manipulation of metadata are present on a number geospatial data dissemination sites. Usually the metadata is entered into an HTML form or Java-programmed[8] interface and subsequently added to the metadata catalogue. However, most of these forms are far too large to be of any practical use for on-line metadata entry given the constraints prevalent with Internet performance and the client's traditional reluctance in completing large Web-based forms.

The retrieval and update of metadata information is poorly implemented in most current on-line cataloguing systems. One primary impediment to on-line metadata cataloguing sites is that most do not allow direct access to the relational database in the form of four primary SQL[9] commands which allow the geospatial data-providers to select, insert, update and delete from their specific database rows over the Web. A number of sites allow geospatial data-providers with adequate means to enter manually metadata to an on-line catalogue. However, on-line facilities to select a specific row and column from the database and change or delete its contents are often lacking from metadata cataloguing sites. As a result, the extremely difficult task of updating and maintaining the database is left to the site administrator who acts upon e-mail correspondences with the data-providers before changing a particular database row and column.

A number of metadata cataloguing implementations do not advocate a Web-based solution to metadata entry. Such systems rely on the distribution of standalone applications or metadata editors that allow data-providers to write metadata into pre-defined fields provided by the application. The results are saved in a proprietary document format which is continually opened, edited and saved, much like a word-processing document, until the completion of the entire metadata file. The proprietary document is then parsed and entered into an on-line metadata cataloguing system thereby

[8] Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded and dynamic programming language (Sun Microsystems, 1995).
[9] Structured Query Language (SQL) is a standardised language used by all relational database management systems that specify the commands to interact with the database.

providing the information to the wider geospatial data-user community. The main problem with this approach is that the metadata editors must be distributed to the user-community at large, supported on a variety of different computer platforms and continually distributed in the event of periodic updates. This is a lot to ask considering the data-provider's traditional reluctance to add new software to their system because of the problems associated with software installation, maintenance and eventual removal. If data-providers wish to submit their metadata to more than one on-line metadata catalogue then they face the prospect of downloading and installing a number of different metadata editors.

Exchanging information between different GIS packages is often a difficult process because of the interoperability issues of geospatial data designed to operate on only specific software systems. Interoperability is a characteristic of computer systems that describes the degree to which the exchange of information occurs with other computer systems. The following questions examine how software, designed to operate over the Web, addresses the problems with exchanging information between different GIS packages through the conversion from one data format to another. Also examined is how such utilities can be integrated into a larger system consisting of many different types of information and processes.

Is the Web able to help address the problem of interoperability between the data formats of different GIS software packages?

Incompatible geospatial data formats are the most prevalent cause of interoperability between the systems of different GIS vendors. The geospatial data formats used in most GIS packages today are proprietary in nature; the formats are designed by a specific GIS vendor and implemented for use within their own family of software products (Fisher, 1991; Flowerdew, 1991). These formats are not compatible with GIS packages from other vendors unless they are converted to a data format specified for the system in question. There are numerous data converters or translators to facilitate interoperability

between GIS software. Most vendors provide such conversion programs with their software but frequently fall short of incorporating all the formats that currently exist in the industry. Also, new data formats are continually being developed, forcing the users of present GIS software into the very expensive option of upgrades. What about those users who do not have access to off-the-shelf GIS packages and require geospatial data for use in some third-party software? Purchasing an entire GIS software package for just a few uses seems a rather expensive option. There are freeware data-conversion utilities that are downloaded from the Internet. Frequently, these utilities have limited documentation, are difficult to install, and difficult to use and may not run on the required platform.

Is it possible to integrate metadata, geospatial data and GIS software using a Web search-engine?

The development of search-engines that access a database, examine retrieved information and use it within on-line software greatly enhances the integration of metadata, geospatial data and GIS software over the Web. The ability to query metadata files from a Web site on the basis of date, keyword and/or co-ordinate information using HTML forms or Java applets[10] is long established. However, despite recent advances in the development of Internet search-engines, the limited search capabilities currently available in most geospatial data dissemination sites are not sufficient for on-line databases (distributed or otherwise) that contain thousands or tens of thousands of metadata files. Because of the considerable variation and complexity of geospatial data, effective search-engines must be able to query on a large number of metadata elements. Also, the software should be relatively intelligent enough to interpret the queried results, suggest the most optimum locations to access the required geospatial data and be able to link the metadata information and geospatial data to appropriate software located anywhere on the Web.

[10] Applets are programs that run in a Web browser when an HTML document containing the applet is loaded into the browser (Deitel and Deitel, 1998).

Web robots are computer programs designed to locate automatically information on the Web and store it within a centralised and conveniently accessible location (Genesereth, and Ketchpel, 1994). They are software applications designed to operate independently by looking for information contained within HTML documents that reference one another forming the inter-networked architecture of the WWW. This portion of the commentary examines how this relatively new technology can be used to accumulate and consolidate GIS related information on the Internet.

How do robots significantly aid in the access of metadata/geospatial data by considerably reducing the amount of time users spend in searching the pages of one or more Web sites?

With the expanding size of the Internet, indexing and cataloguing on-line geospatial data and/or metadata is increasingly being left to Web robots. In many instances, the metadata that accompanies geospatial data references not just one data set but an entire Web site related to a particular topic or class of information. For example, geospatial data from a single geographic area may consist of many different data layers each describing the spatial extent of different attributes. It makes sense to use a single metadata file to describe all of these different spatial data layers but, from an on-line data access viewpoint, this creates a problem. In order to access the data the user needs to search the Web site manually and download the information a single file at a time. For sites that contain a few geospatial data sets this hardly poses a problem, but what of those sites that contain hundreds or thousand of data sets? Such intelligent automated software seems a viable solution in helping users of on-line geospatial data and metadata to explore efficiently potentially vast quantities of information.

The proliferation of GIS software is due to its increasing importance as an essential tool to support decisions made within organisations. The growth of the Internet provides new opportunities in distributing the data used in current GIS software packages to a potentially vast number of users. Existing computer systems possess some key

limitations in sharing geospatial data between users and organisations across distributed networks. At present, there is no unanimity among data-providers for the description, storage and query of geospatial data. If an on-line database is used to store metadata, then what is an effective means to maintain this information? There is no guarantee for those GIS users who access on-line geospatial data, to receive the information in the desired format. Also, it is a daunting task to find and catalogue geographic information from an Internet that increases in size exponentially, without the use of automated software tools. The next section outlines a number of requirements for the development of software designed to facilitate the distribution of geospatial data and metadata over the Internet.

1.3 Specific objectives

The following objectives attempt to address the research issues that affect the searching of metadata and the access of geospatial data over the Web.

1. The primary objective of this thesis is to explore issues concerning the 'Web-usability' of geospatial data by implementing a prototype software system on a Web server. The system is called the Geospatial Metadata Server (GMS). It is designed to locate, retrieve and store geospatial data and metadata. In particular, it aims to incorporate a number of specific features that address particular issues that are currently problematic. These are:
   - The adoption of standardised metadata so as to enable on-line data dissemination sites to integrate indexing and query capabilities amongst themselves. Such a feature ensures that data-providers and users share a mutual understanding for the description of geospatial data.
   - A DBMS to conduct efficient and flexible database queries in order to catalogue, index and search significant quantities of metadata on-line.

- The ability to undertake metadata file parsing and direct loading of information into the columns and tables of a relational database.
- Facilities to support the manual data entry, update and/or deletion by data-providers of their own metadata records within an on-line metadata cataloguing system.
- A search-engine that performs location, date and keyword queries on the metadata database. Metadata retrieved from a query, such as URL[9] addresses, is used by on-line data-conversion software to download geospatial data sets and convert it to a desired format.
- The inter-conversion of pre-defined geospatial data formats by on-line data-conversion software.
- An automated software agent that searches for links to geospatial data from any specified Web site or in response to a search-engine query.

2. A second objective is to evaluate the problems, procedures and issues highlighted by the implementation of this system so as to suggest strategies that might influence and improve the further implementation of systems designed to shared geographic information over the Internet.

3. A third objective is to demonstrate the Web's capability to access on-line metadata and geospatial data by addressing some of the technical and institutional issues that arise as a result of adopting the Internet as new GIS architecture.

The components of GMS (Figure 1.1) are developed to address the research questions outlined in section 1.2 of this chapter. Once developed, the Web-based system is evaluated through a critical review of the operation of each component as well as

[9] A Uniform Resource Locator is an address for a network resource that takes the generalised form of: <protocol>://<host>/<path> The 'protocol' is the method used to retrieve the resource. The 'host' is either a name or numeric notation of a networked server. The 'path' is the networked resource associated with the URL, which is represented one or more subdirectories and a filename (Berners-Lee, 1992a)

through the feedback from users of the system located throughout the Internet. The results are discussed and conclusions are drawn by placing them in a wider context with respect to current and future technological trends within computing industry.

Overall, existing technologies such as an HTTP server, a DBMS and a GIS are merged into a unified system that facilitates the on-line storage and modification of metadata as well as the access and conversion of geospatial data over the Internet. The system stores standardised metadata, which can be entered and/or modified using an on-line metadata editing tool or a parsing program. The metadata stored in the database references geospatial data sets existing throughout the WWW. It can be queried using a search-engine. The results can be fed into conversion utilities capable of accessing on-line geospatial data sets and translating them to any supported format.

As shown in Figure 1.1, the metadata database forms the core of the proposed system. Those components above it are concerned with entering and modifying metadata within the database. Users of such functionality include site administrators as well as metadata data providers. Site administrators are those people who maintain one or more metadata databases by ensuring the information contained therein is organised, current and accurate. They do this by either accessing the metadata database directly or use the metadata editing and parsing utilities provided by GMS. The site administrator must also deal with problems associated with all the software components as well as any enquiries from data providers and data users. Data providers are people that enter metadata into the metadata database and maintain their own metadata records describing geospatial data sets located on their Web sites. The tools data providers use to perform these tasks include the GMS metadata editing and parsing components. They can either be a group of individuals or organisations selected by the site administrator or consist of anyone on the Web who can be invited to contribute to the metadata database.

Those components below the metadata database are used to query metadata from the database and process the information using server software and report facilities. The wider geospatial data user community uses these GMS components. They use the search-engine or Web robot to search the database for metadata and use the information

to access geospatial data of various themes located throughout the Internet. Once accessed, the geospatial data can be converted to another desired format or plotted to a selection of graphics files.

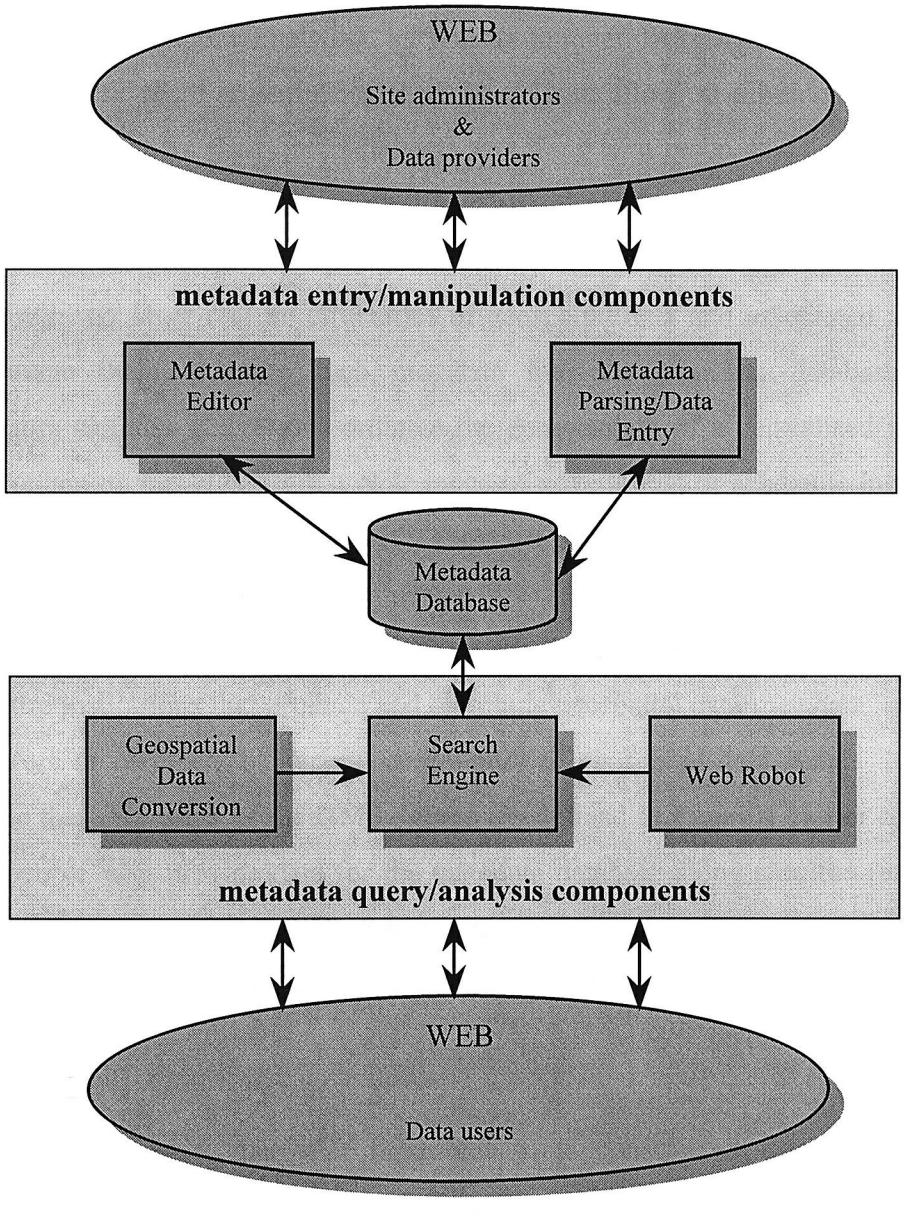


Figure 1.1 The core of GMS is the metadata database component. It is the local repository for metadata information collected and stored from the Web. There are five processing components linked to the database. The editing and parsing functions of GMS are collectively called the metadata entry/manipulation components of the system.

The query/analysis components consist of the data conversion, search-engine and Web robot functions of GMS.

For example, a GMS system set up on a single server can function as an on-line system that serves a private corporate network, such as an Intranet. Alternatively it can be used as a tool by one or more government or academic institutions to provide an extensive catalogue of metadata for their geospatial data sets along with processing tools to analyse and view the data. The system can be operated effectively by a relatively small number of people or site administrators. They can contact perspective data providers who may, in turn, administer large on-line directories of geospatial data and associated metadata and encourage them to enter such metadata into the metadata database. Site administrators can also actively co-ordinate the construction of a centralised metadata database containing metadata from a wide range of government or academic institutions that agree to participate in such a project. They can work with other site administrators to exchange metadata between their perspective GMS systems thereby forming a network of metadata servers located throughout the Web. Therefore, data providers can include entire institutions that produce and distribute geospatial data/metadata, other site administrators or members of general community of geospatial data users. It is also conceivable that data users of the GMS system (as defined in Figure 1.1) can consist of the same people who provide data for the metadata database along side the rest of the geospatial data user community. Therefore, site administrators, data providers and data users are not mutually exclusive categories for people who maintain, contribute and utilise GMS services.

The metadata database, thereafter metadatabase is constructed using a relational DBMS. A relational database (RDBMS) specifies the generic structure of objects of interest as entity sets that are represented as rows within tables and the attributes of the entity sets are represented as the columns within tables (Gittings and Healey, 1989). The relationships between the entity sets are represented as *relations* between tables. The

database schema that specifies the column and table names as well as their relationships is constructed using a portion of the CSDGM standard. Because of the complexity of the standard and the time constraints placed upon the project, only a portion of the standard is used to create the metadatabase. The exact metadata sections and elements used to create the metadatabase are described in chapters to follow. The on-line relational database provides considerable versatility in constructing all the subsequent applications contained within the system, particularly those designed to index and query large amounts of on-line metadata. Through this implementation, the database schema and its on-line database can demonstrate its utility to a variety of Web applications and shows that current relational database technology can be used as a viable storage/query mechanism for complex standardised metadata.

As shown in Figure 1.1, the metadata database incorporates a metadata parsing and data entry utility. Its purpose is to extract metadata automatically from files at remote Web sites and organise this data correctly within the database; it consists of two programs. The first program parses metadata files and the second program loads the values of each metadata field into the appropriate columns of a database record. The parsing program scans a metadata file, character by character, and parses out the relevant sections, elements and element values and stores the results in a text file. The data-entry program reads the text file, creates a new database record and then enters the metadata into the appropriate columns of the metadatabase. The utility is one of the first attempts to enter large amounts of metadata on-line via algorithmic parsing and automatic database entry. The software attempts to demonstrate to geospatial data-providers and administrators of on-line metadata cataloguing sites that large quantities of metadata can be (relatively) quickly and easily entered into a relational database. As the amount of geospatial data and associated metadata increases on the Web, the importance of such metadata cataloguing tools as an alternative to the laborious manual entry of metadata becomes apparent and necessary.

The metadata editor in Figure 1.1 consists of a series of programs that allow data-providers to manually select, insert, update and delete metadata on-line from any metadatabase record or entity, which is subject to proper user authorisation. A row number that corresponds to a particular record in the database designates each metadata file. Security is introduced by controlling access to particular rows with a username and password. Database rows are added or subtracted, viewed and/or changed on any of the metadatabase tables of a particular metadata record. The resulting implementation allows data-providers to manually enter, modify or remove their personal metadata records from an on-line metadata cataloguing system. This level of metadata access and control is unique among metadata cataloguing sites that implement relatively large and complex metadata records. It is hoped that the manual access utility for the metadatabase demonstrates that it is an effective tool in providing direct control to data-providers over their on-line metadata records.

The on-line geospatial data-conversion utilities shown in Figure 1.1 are capable of inter-converting between the data formats of different GIS software packages. The conversion utilities are able to upload geospatial data from any Internet site, convert it to a specified format and return it to the user for download via a hypertext link. Conveniently and pragmatically, only a limited number of data-conversion utilities can be implemented within the system. These data converters or translators are drawn from the data-conversion commands supplied with the Arc/Info GIS software package. The translators for use within the system are capable of converting to and from two specified geospatial data or interchange formats. What is unique about these on-line data-conversion utilities is that traditional data-conversion technologies as seen with the Arc/Info commands are merged with current Internet technologies in order to demonstrate the viability of performing such complex operations over the Web. The entire system forms a 'data-switch-yard' where various GIS formats are converted to a neutral data format and converted once again to another desired data format.

Using URL addresses to reference the location of a specific data file or files, it is possible to access and use geospatial data residing on any connected Web server. The conversion utilities are a clear demonstration that GIS functionality need not be restricted to the local database and server where the software resides. Rather, such GIS applications are able to access data beyond the confines of the local server from a seemingly limitless virtual network of databases.

The GMS search-engine not only queries CSDGM metadata from the database but also examines the information and suggests any on-line Web applications (conversion utilities or Web robot). In other words, the search-engine does not just retrieve information from a database but also uses the information by sending it to on-line applications as arguments on a command line. In most search-engines the user simply queries metadata describing various on-line geospatial data sets including their location on the Internet. The GMS search-engine looks at certain key metadata values, such as the name of the data file and its Internet address. The information is sent to any on-line Web application, say for example the conversion utilities, which uses the file name and Internet address to download the data set and convert it to a desired format. The user is prompted with one or more HTML form buttons containing the metadata information (file name and Internet address), which when pressed, executes the conversion utility on the referenced geospatial data.

In this sense, the GMS search-engine integrates the on-line data-conversion utilities, the Web robot and the metadatabase. These disparate Web applications are no longer separated from one another but are in fact, tied together using the search-engine and the standardised metadata retrieved from the database. Notice also, that these Web applications do not need to be on the same server. The search engine residing on one server can retrieve metadata from a database residing on a second server and send the information to an on-line software application residing on a third server. The search-engine becomes the 'glue' that ties these three components together providing distributed

GIS functionality using only a standard Web browser. The Internet can be used to distribute the storage, processing and display of geospatial data and metadata.

To contend with geospatial data dissemination sites containing an exorbitant number of Web pages as well as the phenomenal expansion of the Internet, the proposed system can be equipped with a Web Robot. Geobot, as it is known, is capable of scanning a site domain, HTML page by HTML page, for links to geospatial data using their file extensions as identification markers. The resulting geospatial data links are either entered into a database to be queried much like other Web robot sites or the information is sent to the user in the form of an HTML document showing each geospatial data link and its page of origin. Geobot constitutes among the first attempts to search for hypertext links to on-line geospatial data using automated WWW software. It is hoped Geobot demonstrates the use of Internet Agents in automatically finding and cataloguing very large Web sites containing geospatial data sets and their associated metadata.

1.4 Thesis structure

The development of the World Wide Web since its inception in 1991 and its propagation thereafter provide the means to merge a variety of technologies to facilitate the exchange of on-line geographic information. The purpose of this thesis is to examine the design and implementation of GMS and suggest methods to improve the search, catalogue, maintenance and processing of metadata and geospatial data over the Internet. The rest of this thesis is presented in the following manner. Chapter 2 examines how contemporary Web implementations for accessing on-line metadata and geospatial data address the research issues discussed in the present chapter. Chapter 3 and 4 takes an in-depth look at the development of a GMS prototype system by examining the components responsible for metadata entry/modification in the former chapter and metadata query/analysis in the latter chapter. The programming code for the proposed prototype is

located on CD-ROM in appendix A. Chapter 5 discusses the implementation, performance analysis and user evaluation of the software prototype in light of what each GMS component is designed to achieve, the objectives it meets and problems it solves. Finally, the main conclusions of this research and future research are presented in chapter 6.

CHAPTER TWO

GIS IN THE INFORMATION AGE

The spread of Web technology since its inception at the beginning of this decade has facilitated a number of important developments within the GIS industry. These developments have recently made GIS technology far more accessible to a large and increasing number of people. In fact, Web technology extends the more traditional uses of networking technologies by centralising services that are accessible by 'smart' clients. Systems that are designed to access, process and display geospatial data over the Web have been in use for approximately six years. Such systems are steadily improving in their ability to store, process/analyse and present geographic information over the Web. This also includes the ability to analyse and display maps, tables, images, diagrams and other related information on-line. With the advent of mature standards, robust servers and more powerful development tools, very useful functionality that exploits the full potential of geospatial information is now available over the Internet.

This chapter examines a broad range of functionality of the technology comprising currently implemented geospatial data dissemination, metadata cataloguing and mapping Web sites. Due to the extensive nature of the field of study, the commentary is limited to only those systems or portions of systems that bear direct relevance the Geospatial Metadata Server (GMS), which in turn, elucidate the issues addressed by this research. Plewe (1997) classifies entire systems and the services they provide into a number of distinct categories derived from increasing system complexity. For example, some geospatial data dissemination sites provide only directories of geospatial data sets; other sites only process data while others perform GIS analysis and display operations. However, this method of observing entire geospatial data dissemination sites does not account for systems containing all or

most of these services. Examining the functionality of such diverse Web-based systems is more useful than simply concentrating on system complexity. GMS is divided into six different components that are functionally distinct from one another. The following chapter examines these different types of functionality within separate sections. This is achieved by critiquing comparable functionality within contemporary on-line geospatial data dissemination sites in order to understand the issues and concepts that underpin the aims of this thesis.

2.1 Metadata and Information Discovery

The Web allows, for the first time, a large number of users within the GIS and Remote Sensing communities to access a variety of different types of geospatial and earth science data. The Web sites that contain this data and the software designed for their distributions rely on standards of data description that are well established in the industry (Healey et al., 1998). Metadata, as it applies to GIS, can be defined as information or data describing one or more geospatial data sets. Such a description usually consists of a formal specification or document about the digital data, categorising the document into sections, which, in turn, are broken down into descriptive fields (Schweitzer, 1996a). In many cases, this is done on an ad hoc basis but there has been considerable effort within the spatial data-user community to negotiate and develop national and international standards for metadata. A number of different standards are in use, some of which are discussed below.

Because geospatial data is such a substantial investment for most organisations, the use of a metadata standard allows an organisation to manage effectively substantial quantities of their data. This results in a better utilisation of resources and an improvement in overall efficiency. Metadata provides the means to catalogue geospatial data and assemble it into vast clearinghouses or distributed databases (Nebert, 1995a; Frank, 1994). Data from external sources can be merged seamlessly with existing holdings that enhance an organisation's ability to acquire and use resources that exist beyond its operational control. Ultimately, metadata allows

organisations to determine the availability of geospatial data for a particular geographic location. Metadata also provides data quality information that is necessary to determine if a data set fulfils a specific need or has been produced to a particular standard. Usually, vast collections of geospatial data are catalogued with a single metadata file, which, in turn, provides information to access the entire collection of data. Information regarding the transfer of geospatial data (method, mode, file format) is an important part of a metadata record. The information contained within metadata records can be accessed sequentially, recursively or randomly depending on the required needs of the end user.

2.1.1 Metadata standards

The types of metadata outlined in this section are composed of standardised attribute/value pairs. The attributes in the term attribute-value pair are individual characteristics of metadata information (i.e. metadata elements mentioned below). The values are the actual metadata information associated with these metadata characteristics. An example of an attribute-value pair is Network_address:http://www.geo.ed.ac.uk.

Producing standardised rules for the use of metadata requires the overall consensus from the entire user community; this is not at all straightforward. The current CSDGM standard is one such example of this category of metadata (Federal Geographic Data Committee, 1994). Essentially, attribute/value pairs that consist of metadata elements, compound elements and associated values are defined by select committees producing successive draft documents. These documents eventually become formalised into a final national or international standard used throughout the spatial data-user community.

2.1.1.1 The CSDGM standard

At the start of this project, the Content Standard for Digital Geospatial Metadata (CSDGM) was by far the most developed metadata standard available both for its familiarity within the GIS community and the number of geospatial data sets it describes on-line. The CSDGM standard provides a common set of terminology and

definitions about the content, quality, condition and other characteristics of digital geospatial data. The final version is more exactly called the June 8th 1994 CSDGM standard[1]. A description of the most prevalent structural features of the standard is given below. It is important to explore this because of its central role in the development of a number of GMS components.

The standard is organised in a hierarchy of data elements and compound elements that define the information content of metadata as a means of documenting a set of geospatial data. Metadata elements define the specific characteristics of digital data. For example, *Online_Linkage* is a metadata element defined in the CSDGM standard that references an online computer resource as a URL address. Similar elements can be grouped together to form compound elements to further clarify the metadata document and enhance its interpretation. For example, *Browse_Graphic* is a compound element describing a binary graphic image that provides an illustration of the data set. This compound element is composed of three individual elements *Browse_Graphic_File_Name, Browse_Graphic_File_Description,* and *Browse_Graphic_File_Type*. The elements and compound elements are placed in numbered sections and subsections that divide the standard into descriptive categories. The sections, subsections, compound elements and elements form hierarchical relationships between one another. This controls the level of abstraction when describing a geospatial data set. For instance, high level abstractions can be maintained at the section/subsection level while progressively lower levels of abstraction are defined by compound elements and elements.

The CSDGM standard is an extensive and comprehensive means to document geospatial data. This section explains in further detail the structure of the metadata standard as it applies to the development of the GMS metadatabase. Generally, there are 334 different elements in this standard of which 119 are compound elements that are composed of other compound elements in a hirarchical structure representing

[1] The final draft of this standard was released between January and March 1994 by the Federal Geographic Data Committee (FGDC) after extensive reviewing and testing by potential users. The FGDC is both the approval and maintenance authority for the CSDGM standard. The standard is also being submitted to the US Department of Commerce for approval as a Federal Information Processing Standard. The standard was later revised in 1998 and named accordingly as the 1998 version of the CSDGM standard (also see FGDC, 1994 and FGDC, 1995).

various aspects about the data set (Schweitzer, 1996a). These elements are divided into 7 major sections that include (Federal Geographic Data Commitee, 1994):

- *Identification_Information* includes basic information about the data set such as its citation and time, which are both contained in separate sub-sections. The section also contains G-Polygon co-ordinates for delineating the outer area of the data set, keywords/keyword types, access/use constraints, and information on graphic representations of the data set and more.
- *Data_Quality_Information* section contains a general assessment of the quality of the data set, which includes an attribute accuracy report, a logical consistency report and a completeness report. In addition the section includes information on positional accuracy of spatial objects within the data set, lineage or source data and percent cloud cover if relevant. See Burrough, 1986 and Burrough, 1992 for an in-dept analysis of issues concerning data quality and its implications towards GIS.
- *Spatial_Data_Organization_Information* contains information on the mechanism used to represent spatial information in the data set. Some of its more important elements include the system the data set uses to represent the objects in space (i.e. point, vector or raster) as well as detailed information concerning each of these spatial reference methods.
- *Spatial_Reference_Information* contains the description of the co-ordinates in the data set in terms of its reference frame and encoding method. This section includes information such as map projection, latitude/longitude resolution, geographic co-ordinate units and so on.
- *Entity_and_Attribute_Information* contains information about the information content of the data set, including the entity types, their attributes, and their domains from which attribute values are assigned.
- *Distribution_Information* contains information about the available format of the data set, its medium of transfer, and contact details of its distributor.
- *Metadata_Reference_Information* contains information about the metadata file including its creation date, its review date, the metadata standard it follows and its creators.

Citation information, temporal information and contact information are used throughout the metadata standard. Because they form sub-sections contained within some of the larger sections described above, the designers of the standard felt it prudent to contain all such information (wherever it is used) within its own accessible metadata section.

- *Citation_Information* contains the recommended reference to be used for the data set.
- *Time_Period_Information* contains temporal information in the form of date and time and is used throughout the metadata file.
- *Contact_Information* contains information on the identity of the person(s) or organisation(s) associated with the data set and the method to contact them.

For example, both the *Data_Quality_Information* and the *Identification_Information* sections of the CSDGM standard use citation information. Therefore, citation information need only be defined once within the CSDGM standard for those sections. Similarly, *Identification_Information*, *Data_Quality_Information* and *Distribution_information* sections all use temporal information defined by the *Time_Period_Information* sub-section. Also, the *Metadata_Reference_Information* and the *Distribution_Information* sections define contact information using the *Contact_Information* sub-section. The CSDGM standard specifies certain mandatory fields, which can be construed as a minimum subset to be completed for each metadata record. The optional metadata fields support the information contained in the mandatory fields by addressing the considerable variation of geographic information as represented by different sets of geospatial data.

2.1.1.2 The DIF standard

The Directory Interchange Format (DIF) is another important attribute/value pair metadata standard. This is used extensively for data description for all data sets contained within the NASA sponsored Global Change Master Directory (GCMD). DIF is a de-facto standard used to create directories in the form of database entries describing a group of data or data sets. GCMD is a good example of using a metadata standard as a means to describe and distribute earth science data over the Internet. An overview of the DIF structure is given in this section because a number

of GCMD system components are reviewed in this chapter that requires some basic knowledge of this standard.

Data is described by DIF through a series of fields with some containing text and others numeric values. Six of these fields are required for each data record while optional fields are available for further clarity. The required fields include.

- A *Directory Entry Identifier* specifying a unique identifier for the DIF file.
- A *Directory Entry Title* specifying a title for the DIF file.
- A set of *Parameters* representing various measurement of the data such as Category, Topic, Term, Variable, Detailed Variable.
- An *Originating Centre* specifying a short name of the data centre that originally assembled the data set.
- A *Data Centre* specifying information on the data centre that distributes the data such as its URL address, data set identification, and contact details.
- A *Summary* containing an abstract, information not contained in other fields and the overall usefulness of the data set.

DIF is also capable of describing geospatial data because the Federal Geographic Data Committee (FGDC) has required the addition of seven more fields to comply with some of the fields in the CSDGM standard.

2.1.1.3 USMARC formats and Discovery metadata

The USMARC formats are standards for the representation and communication of bibliographic and related information in machine-readable form. A USMARC record involves three elements: the record structure, the content designation, and the data content of the record. A format is a set of codes and content designators defined for encoding machine-readable records. They are defined for five types of data: bibliographic, holdings, authority, classification, and community information. The Library of Congress maintains the USMARC formats in consultation with various user communities.

Many of these formalised geospatial data standards are actually based on previously developed standards. CSDGM along with future international metadata standards are no exception, which explains why many attribute/value pairs tend to be similar across the development committees of different standards. For instance, USMARC

is a metadata standard used by the library community to facilitate the exchange of catalogue records that employ the *Anglo-American Cataloguing Rules (AACR2)* for describing many types of resources. The structure of USMARC records is based on the Bibliographic Information Interchange (ANSI Z39.2, another library cataloguing protocol like Z39.50) and Format for Bibliographic Information Interchange on Magnetic Tape (ISO 2709). The data elements for a record that specify how data are manipulated are defined in USMARC formats. The content of most data elements are defined either by other standards (Anglo-American Cataloguing Rules, Library of Congress Subject Headings, National Library of Medicine Classification) or by USMARC formats such as in the case of coded data (Machine-Readable Bibliographic Information Committee, 1996).

Another metadata standard that requires an introduction is the Discovery metadata guidelines which where developed by the UK's National Geospatial Data Framework (NGDF, 1999). It provides a consistent and simple method of documenting and cataloguing geospatial data particularly with respect to data resources accessible via the Internet. It is far simpler than the CSDGM standard with only 42 metadata elements of which 16 are mandatory, 7 conditional and the rest are optional. Therefore, the standard can be limited in describing various types of geospatial data sets as well as their level of detail. The guidelines are based on the draft ISO metadata standard 15046-15 which is actually similar to the FDGC CSDGM standard discussed in sub-section 2.1.1.1. Along side ISO and CSDGM there is the Open GIS Consortium (OpenGIS®, 1996) which consists of a collection of organisations that are striving to define standards for interoperability in GIS. There is a considerable amount of overlap between all these organisations. Hence the terminology used in their standards is often similar.

2.1.2 Information discovery protocols

Some of the geospatial data dissemination sites examined in this chapter use standardised metadata along with information discovery protocols. These application-layered protocols operate in a distributed-networked environment. This provides not only on-line data description but also information management,

catalogue and search services across a multitude of clients and servers. The protocols discussed in this sub-section use different approaches to handle metadata information. They represent metadata with:

- no explicit structure;
- open ended definitions;
- standardised collections of attribute/value pairs.

These points form a loose classification (adapted from Cuthbertson et al., 1994) based on the metadata structure and its implementation on both client and server applications. The sub-sections that follow explain each category, provide examples and review their use in accessing on-line information.

2.1.2.1 A system management protocol

The 'finger' or 'whois' protocols usually return metadata to clients that are dependent upon their implementation on a particular server. The metadata that is returned as part of a response from one finger/whois server is not necessarily the same for other server configurations. As a result, the metadata is said to be void of any explicit structure because of the differences in their implementation from one server to another. Whois is a protocol for finding information by sending requests to servers, which returns information about the organisation, administrators, contact information and IP address. Whois is similar to the finger protocol except that the former runs on a certain number of servers while the latter is implemented on most servers. Both protocols can be used to design on-line distributed Web applications. For example, the 'finger' protocol plays a crucial role in the World-Wide Earthquake Locator (Towers and Gittings, 1995). See section 2.5.3 for commentary on how the 'finger' protocol is used in this system.

2.1.2.2 A directory service protocol

The X.500 protocol implements the Open Systems Interconnections (OSI) model for connecting local directory services to form one distributed global directory. Such a configuration decentralises maintenance of the directory because each site is responsible for only its portion of the directory. Local databases hold and maintain a

part of the global database and the directory information is made available via a local server called a Directory System Agent (DSA). The user perceives the entire directory to be accessible from the local server (Weider et al., 1992a; Weider et al., 1992b). While most of the information available today via X.500 is about people and organisations, the design of the X.500 directory is also suitable for storing information about other entities (or objects), such as network resources, applications or hardware.

The X.500 server allows the combination of multiple object classes. Each *entry* in the X.500 directory describes one object (e.g. a person, a network resource, and an organisation). The entry consists of a collection of *attributes* (e.g. for a person this might be last name, organisation name, e-mail address). Therefore, each entry is defined by a number of attributes-value pairs consisting of an attribute type and its value. An attribute can have multiple values. An entry can have both mandatory attributes and optional attributes. Each entry has a single attribute known as an *objectClass,* which lists the set of required mandatory attributes and optional attributes in an entry of that class. The object classes are arranged in a hierarchical manner according to class inheritance. The object class concept used for X.500 allows one to create new object classes from previous classes. This is an example of metadata with open-ended definitions. Existing objects can be combined to form new objects capable of describing a particular type of information. For example, given a collection of objects (organisation, person, number, mail, and phone), new attributes can be created by combining one already defined such as organisation-number, person-mail or phone-number.

2.1.2.3 A database search protocol

The Z39.50 is an application-layer protocol for network communication between heterogeneous hardware/software systems, multiple database searches and information retrieval[2]. Implementations of the protocol use a common interface

[2] *ANSI/NISO Z39.50-1995 - Information Retrieval (Z39.50): Application Service Definition and Protocol Specification.* The standard was developed by the National Information Standards Organisation (NISO) which is charged with maintaining standards for information services, libraries and publishers. NISO is an accredited organisation by the American National Standards Institute (ANSI). Z39.50-1995 will eventually become an international standard for information retrieval (Z39-50, 1995).

allowing information providers to integrate many different databases and servers without having to rationalise any of their hardware and software systems. The protocol is based on the client/server model. Z39.50 is specifically used for library applications to provide a facility to search multiple databases without the need for the users to acquire knowledge specific of any particular system or database (Naik, 1987). Unlike the Web, the Z39.50 protocol is capable of performing a query by obtaining additional information from the user in multiple steps. This means that the protocol has 'state' because of its ability to remember previous queries. However in order to achieve this ability a connection must be established to a Z39.50 server and maintained over the entire time information is transferred (Z39-50, 1992; Z39-50, 1995).

The Wide Area Information Server (WAIS) is a full-text informational retrieval system that implements the Z39.50 protocol for locating information across a variety of databases using natural language queries. WAIS directory servers are used for the National Geospatial Data Clearinghouse (NGDC) forming an archive of metadata information, which is distributed among many servers and accessible to users through a single point-of-entry or gateway (Nebert, 1995b). The gateway software developed for the NGDC allows the clearinghouse's Z39.50 servers to be accessible over the WWW using common Web browsers. In this context, the protocol standardises the messages exchanged between the client and server software as well as the semantics of the search query. For example, the protocol defines a table of attribute semantics or search-terms known as an *Attribute Set*. The 1995 version of the protocol defines the following attribute sets (Hakala, 1996 and Kunze, 1995):

- Bib-1 - Bibliographic-1 attribute set;
- Exp-1 - Explain attribute set;
- Ext-1 - Extended services attribute set;
- CCL-1 - Common Command Language attribute set;
- GILS - Government Information Locator Service;
- STAS - Scientific and Technical Attribute Set (a superset of Bib-1).

Existing attribute sets can be enhanced with new attributes or new sets can be created for new search domains. In spite of its successes, the overall use of the Z39.50 protocol has declined with the advent of purely Web related search and retrieval

methods (McClure and Moen, 1994). This is due to the Web's flexibility in providing vendor specific search-engines to site administrators that are constructed using widely available WWW protocols.

2.2 On-line metadata databases and catalogues

This section is divided into two parts. The first part examines the range of technologies and architectures used for some of the current on-line metadata databases and catalogues. The second part builds upon this commentary by presenting examples of such systems and critiquing their implementation in terms of their system architecture, functionality, effectiveness and usefulness.

2.2.1 Metadata storage and query

Before embarking on a description of the GMS database implementation, it is useful to examine the various methods in use for systems that catalogue and store on-line metadata. The way in which geospatial metadata is stored, whether as ASCII files or within an RDBMS affects fundamentally the access of metadata using server-client systems. In many instances, combinations of methods are used to index and catalogue metadata. Some storage methods are better suited for certain circumstances due to a variety of external factors such as performance issues (query response times, file size, network bandwidth and server load) and user requirements as well as internal factors, which includes the design of the system.

For example, on-line metadata catalogues comprising of flat files allow the user to download complete metadata files relatively quickly even at times of low network bandwidth or heavy server load. However, the information contained within individual metadata files is not easy to access because the excessive length of these files makes it difficult to extract manually particular bits of data. Alternatively, metadata parsing programs when used in the context of browsing and searching on-

line metadata tend to be impractical for large or even medium-size databases because of the excessive amount of processing that must be conducted on relatively slow Internet connections. Despite such disadvantages, a number of major metadata cataloguing sites have designed their systems to operate under such conditions (Ogilvie, 1995).

Geospatial metadata cataloguing sites implemented using an RDBMS not only have considerable storage capacity but also are able to search and query large quantities of information including all aspects of a metadata record. A properly designed database is able to query values from any table or column of a metadata record representing standardised metadata sections and element names, respectively. This facilitates the on-line access of metadata through equally well designed search-engines. Metadata databases are capable of returning a large number of metadata records with a single database query. The capabilities of such a diverse yet powerful software package should not be underestimated because database servers form the foundation of a vast industry of information storage and management. The primary disadvantage of using a RDBMS is that they are large and complex software packages that are considerably more difficult for site developers to master than simply referencing flat files. A significant degree of additional structural complexity in the software is introduced that requires the designer to invest time in setting-up and planning the system. Prior to storing data within a RDBMS, the developer must design a database schema consisting of database tables and columns linked in different ways to one another according to an entity-relationship model[3]. Database queries must also be designed using the SQL standardised query language that acts upon the database according to the specified database schema.

Any number of metadata databases accessible over the Web can be brought together to work seamlessly as a single database capable of being browsed and queried as long as each networked database (accessible via a Web server) shares the same database schema. These networked database management systems can be integrated and decentralised for use to a wider community of users either within a single

[3] A top-down approach to database design that consists of the selections of *entities* followed by the specification of their *relationships* and the assignment of *attributes* to these entities and relationships in such a way that a set of fully normalised tables is obtained (Howe 1989).

organisations, across many organisations or the entire Internet (Orlowska, Lister and Frogg, 1992).

Asking different metadata cataloguing sites, geospatial data-providers and site administrators/developers of geospatial data dissemination sites to use the same metadata standard, implemented using the same RDBMS schema, requires a considerable amount of co-ordination among the concerned parties. In many instances, developers of such sites may in fact use an RDBMS to store metadata but either the metadata is stored in a non-standard and ad-hoc fashion or the database schema used is vastly different from that used by other site developers. The use of dissimilar database schemas (for those developers selecting the RDBMS route to store and query metadata) forces the development of customised Web search-engines, which can not work with other on-line metadata databases. If each site developer uses customised search-engines then this precludes the possibility of accessing multiple on-line databases through a single Web interface.

A number metadata cataloguing and on-line search sites use a combination of flat files for metadata storage and an RDBMS for metadata query to provide access to their databases over the Web. Of course, flat files exhibit some of the same problems associated with on-line metadata storage and query mentioned earlier. However, a degree of conformity can be achieved between disparate on-line metadata cataloguing and data dissemination sites through the construction for each site of a common scaled-down relational database used primarily by the search-engine. In other words, each site stores metadata on its server in the form of flat files and uses an RDBMS to provide citation information of each of its metadata records over the Web, which can then be accessible via a common search-engine or gateway. The gateway combines the metadata records that are held within all participating servers into one large *federated* collection, which is accessible from a single Web address.

A federated database provides a unified view of a collection of heterogeneous databases that are linked in a similar manner. The entire database is designed from the bottom-up, meaning the individual components or Web servers (in this case) are each implemented separately and given a considerable amount of autonomy in their operation. The operation of each server is controlled concurrently by one or more applications located at various locations on the Web, which send SQL query

commands in the form of HTTP (or other protocol) requests to each server and subsequently awaits their responses. The applications (usually search-engines but they can consist of other types of software as well) process the responses received from the on-line federated databases and format the information for presentation to the user. At present, there are no federated databases that use a full RDBMS implementation for both metadata storage and metadata query. However, a couple of systems are discussed in the following sub-section that do come very close.

2.2.2 On-line metadata storage and query systems

Digimap is a good example of a geospatial data dissemination site that uses an RDBMS to store non-standardised metadata information and provides access to the information over the Web. The Digimap Project began with the aim of assessing different strategies by which students and staff in higher educational institutions can efficiently and cost-effectively access Ordnance Survey (OS) digital map data. A working prototype was constructed containing a subset of OS data with subsequent user-trails to test the functionality of the system.

Medyckyj-Scott and Morris (1998) perceived that and important service requirement for the prototype system is an easy to use graphical user-interface (GUI) available to both libraries and end users. The GUI (described in more detail in section 2.6.1) is downloaded from the server as Java byte code and is executed on the client. In order to obtain information from the metadatabase (Figure 2.1), the client simply requests the HTTP server for information and the server builds the appropriate SQL code using CGI[4] scripts. The SQL commands then access the database and the metadata is returned to the client via a HTTP response.

[4] The Common Gateway Interface is a platform-independent interface used to run backend software in conjunction with an HTTP server (Brebner, 1997).

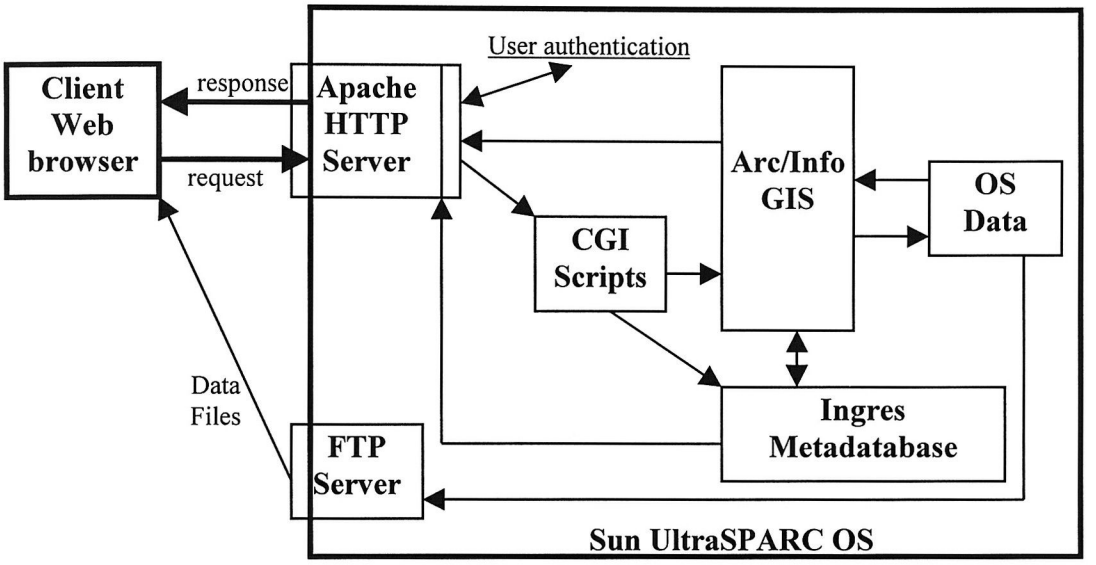


Figure 2.1 The Digimap implementation showing the RDBMS metadatabase and its relation to other system components, in particular the HTTP server (from Medyckyj-Scott and Morris, 1998).

Though the process is simplified in the above diagram, it does demonstrate how functionally-rich user-interfaces, such as the one used for Digimap, can be used to access a metadata database via the Web.

Importantly, Digimap is also scaleable to other library sites. Once the system completes its stages of development, the software can form the basis of a federated system of databases that ties metadata records from numerous library sites. These are capable of being searched from a single user-interface located anywhere on the Web. The use of an ad-hoc method of geospatial data description (i.e. metadata) is a serious disadvantage for Digimap. All the participating library sites must ensure their metadata records adhere to a pre-specified format for them to be stored within the Ingres Metadatabase (shown in Figure 2.1) because all these databases must possess the same database schema. If the database schema were to change for a particular library site, then the CGI scripts controlling the Digimap Apache server would be rendered inoperable, effectively precluding the user from accessing any on-line metadata from that site.

The central importance of metadata is that it allows systems such as Digimap to keep track carefully of information passing between different servers, GIS packages and databases. Future development of Digimap consists of speeding up response-times and distributing the database such that the OS data need not be located at any one particular site. The new software should in future be able to access the OS data from a number of remote sites over the Web. Components of Digimap should also be distributed in order to share the processing power between different computing systems. Software on such a distributed scale requires the sort of interoperability[5] that is achieved using standardised metadata within an RDBMS to describe geospatial data sets.

GENIE[6] is another on-line cataloguing system that uses only flat files as its primary means of storing metadata. These are distributed throughout various network locations in a federated fashion and referenced within a searchable centralised directory. The entire system consists of a federal metadata network for the United Kingdom whose purpose is to facilitate the access of metadata relevant to global environmental research. The system allows Global Environmental Change (GEC) researchers to retrieve and manage data sets that have been obtained from the network as well as reference their own data on-line in order to provide accessibility to other GEC researchers (Genie project, 1995).

The project aims to make available all these disparate types of data either through data centres or individual researchers which reduces the need for any particular researcher to duplicate existing data or to perform analysis that has been completed by other researchers or organisations. Data centres and GEC researchers are linked together with GENIE to the Joint Academic Network (JANET). Interestingly, the project is not modelled around a general client/server model, which precludes its use among the wider Internet community. Each user can implement a GENIE sub-system consisting of its own metadatabase and user-interface for accessing metadata and the system is capable of interacting with other GENIE sub-systems (Medyckyj-

[5] Interoperability is the ability for different systems to communicate and exchange information with one another. See section 2.5.1 for a more complete definition.

[6] The system is developed by academic departments from the university of Nottingham, Loughborough and Leicester with additional contributions from government and private organisations such as Genasys II Ltd.

Scott et al., 1996). This important limitation confines its use to the academic community since only those who obtain and install a copy of the software can use GENIE.

GENIE has the unique quality in comparison to other network metadata cataloguing systems, of using a Master Directory in order to reference metadata files located on participating network sites (that stores accessible metadata files).

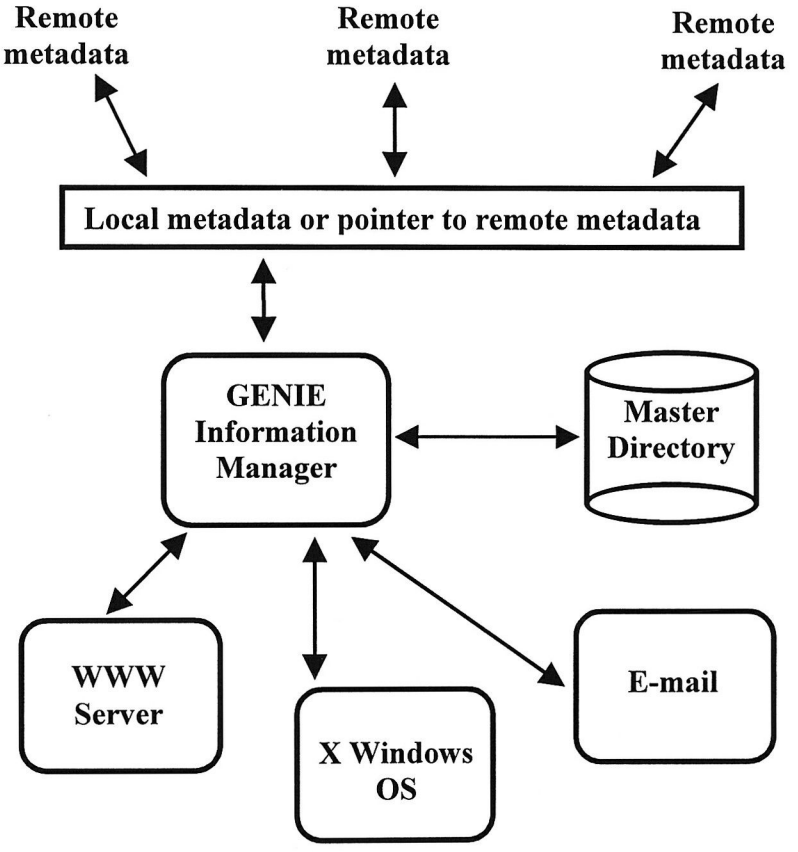


Figure 2.2 Some of the components associated with a GENIE node including the Master Directory designed to reference local and remote metadata files (from Genie project, 1995)

A GENIE node (Figure 2.2) consists of its management software and a master directory containing document titles and keywords that are searchable via the network. Access to any particular data set is through the submission of queries to the Master Directory which are composed of concepts or keywords associated the directory entry to its metadata file located somewhere on the network.

Among the primary disadvantages of a system such as GENIE is its reliance on the Master Directory as a referencing mechanism used between geospatial users and remote metadata files. As is so often the case when using distributed networks, references can become outdated as files are moved, changed or deleted from a particular location. Within a short period of time, the directory can contain a considerable amount of data that no longer exists on the network. Data-providers may have removed their metadata and geospatial data files, renamed the files or simply changed their directories.

The search parameters used for the Master Directory are also limited consisting of a few fields that make it virtually impossible for a search-engine to narrow the search criteria given that a federated database may consist of numerous metadata files. Because the data-providers are required to maintain both the metadata records and the geospatial data files, both files reside together at the same network location. This architecture precludes any GENIE node from having access to the enormous amounts of information contained within metadata records for use within on-line search and query routines.

The National Geospatial Data Clearinghouse (NGDC) is among the largest examples of a nation-wide federated cataloguing system for metadata. The system was constructed as a result of an executive presidential order (see Clinton, 1994; Circular A-16, 1994). It primarily is an early US initiative to co-ordinate the implementation of a national spatial data infrastructure which includes the construction of a national geospatial data clearinghouse where geospatial data described and defined in a standardised fashion. A later European initiative uses the EDIFACT[7] international standard in the construction of an Electronic Data Interchange (EDI) throughout the continent (Rainio and Ahonen, 1994).

NGDC possesses several relatively powerful search-engines, which are used to query standardised metadata from various networked servers. The search-engines send user-queries to different servers, which respond by returning CSDGM compliant metadata of the geospatial data sets currently residing on the system

[7] Electronic Data Interchange For Administration, Commerce and Transport (ISO 9735) is a generic, international syntax for data representation (Rainio, 1995).

(Nebert and Fullton, 1994). The use of standardised metadata is a substantial improvement for such systems because it allows fairly sophisticated and expressive search-engines to be constructed, capable of querying metadata across diverse organisations and data-providers. The search-engines initially constructed consisted of HTML forms that were later updated with Java applets. These Java-enhanced search-engines provide GUI facilities such as panning/zooming of map layers and the drawing of rectangular search areas in order to enter more easily search criteria based on map co-ordinates. The Clearinghouse operates using two network protocols, HTTP and Z39.50, implemented as separate servers on each site. Refer to Figure 2.3 below.

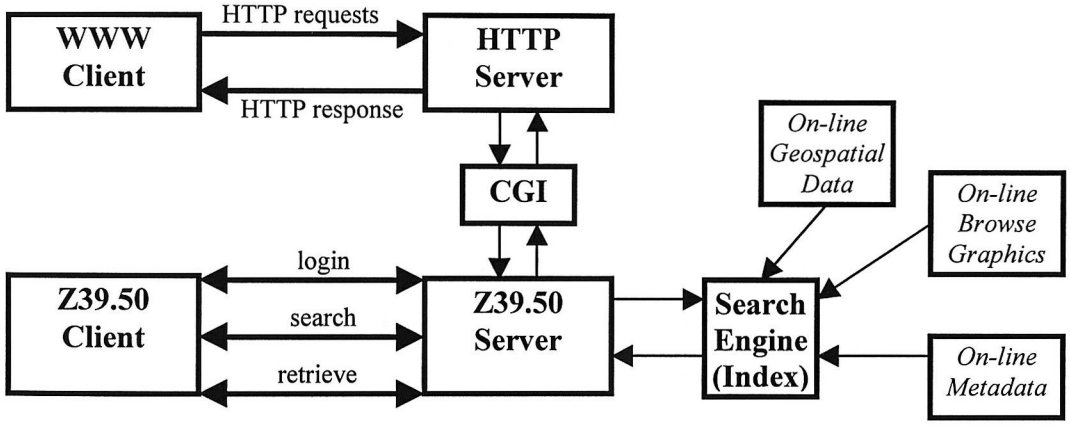


Figure 2.3 Dual server configuration for the National Geospatial Data Clearinghouse (from Nebert, 1995b). Web clients can access the search-engine using the HTTP server via CGI scripts, which, in turn, accesses the Z39.50 server as a backend application. Z39.50 clients can access the search-engine directly using a server that implements the Z39.50 protocol.

The dual server configuration of the NGDC consists a WWW and a WAIS server (Nebert, 1995b). Either WWW or WAIS clients can be used to query their respective servers using the HTTP or Z39.50 protocols. WWW clients can send requests to the HTTP server via HTML forms, which are processed and forwarded to the Z39.50 server by CGI scripts. In essence, the Z39.50 server operates as a backend application to the HTTP server. It is the Z39.50 server that contacts the search-engine in order to access metadata, graphics and geospatial data via URL references from one or more on-line data resources (or clearinghouse participants). Not surprisingly, the HTTP client and server operate on a request-response model

where information is requested and a response returned before a second request is initiated. In contrast, Z39.50 clients must initiate an interactive login session whereby a connection to Z39.50 server must be maintained for the length of the search or query. Because the protocol possesses 'state', the commands used to query the data can be reconstructed continually for the length of the session.

The metadata is stored on various servers throughout the network as flat files alongside their associated geospatial data sets (Figure 2.3). NGDC is designed to keep both metadata and its geospatial data in the same location such that their respective data-providers can maintain both sets of files. The search-engines (which prompts the user to select one or more on-line servers) are constructed on top of an RDBMS containing citation-information, bounding co-ordinates, keywords, date-information and distribution-information search parameters. Such a model requires that portions of the individual metadata files must be copied to the proper tables and columns of the RDBMS in order for associated geospatial data to be accessible over the Web. Any changes to key bits of information or even removal of the metadata/geospatial data file by the data-provider must be reflected in the database. These necessary modifications place a considerable burden on the administrator of the on-line metadata catalogue in ensuring that database records are current and up-to-date. Also, large parts of the metadata files can not be accessible by the search-engine and hence can not be searched by the user because the information is not included in the RDBMS.

The National Geospatial Data Clearinghouse succeeds by providing a referral service for information servers that contain spatial information. This is accomplished using the Web as a universal user-interface and WAIS to provide search capabilities that can run across multiple servers. The US Geological Survey is the first agency within the NGDC to use this technology to demonstrate a distributed clearinghouse that can be browsed and searched of spatial data. The Johnson Space Centre is also using this system to spatially index and serve over three hundred thousand scanned photographs acquired from Earth observing missions. These photographs can be selected from a Web browser by drawing bounding rectangles over an image of the Earth.

The Global Change Master Directory[8] (GCMD) is currently the only on-line metadata cataloguing system that consists of a federated database storing metadata records within an RDBMS. The system does not directly use any flat files to store metadata; however, it does accept such files from data-providers over the Web and subsequently parses them prior to database entry. The metadata describes satellite images and other earth science data concerning the atmosphere, hydrosphere, biosphere, oceans and earth. GCMD was originally conceived to find and view metadata on science-related data across a variety of data servers. The on-line software system addresses the problem of various agencies using different data description methods by developing a directory metadata interchange format (DIF). This directory is implemented on a decentralised network of directories containing metadata across agencies, disciplines and international boundaries (Olsen, 1996).

The DIF metadata standard contains information fields to assist in normalising database searches within GCMD and allows users to locate easily desired data anywhere on the system. The standard is not as expressive as the CSDGM standard used in the NGDC for describing the geospatial data sets. However, the relative simplicity of DIF, particularly in its ability to be parsed via algorithmic means, makes it ideal for use in on-line metadata cataloguing system such as GCMD. The generalised nature of the standard allows it to describe digital data from the diverse range of disciplines that GCMD encompasses.

The database is implemented in the ORACLE relational database management system. It is a very extensive database containing earth science data sets numbering over five thousand directory level metadata records. It is a clear demonstration of the size and complexity of currently implemented on-line metadata databases. A typical GCMD database server is a standard TCP/IP server operating on a 'connectionless delivery' mechanism (Thomas, 1996 and section 2.5.2). Due to the size of the database, the use of a RDBMS to store metadata makes it considerably easier for data-providers to modify their own metadata records rather than having such labour-intensive tasks being placed on the site administrator. Modification of metadata

[8] The National Aeronautics and Space Administration (NASA) developed GCMD to assist the scientific community in querying on-line metadata from a large number of agencies, organisations and subject disciplines (Olsen, 1996). The database was constructed by the Committee on Earth Observation Satellites International Directory Network (CEOS IDN) and GCMD acts as a co-ordinating node of the CEOS IDN by maintaining a complete copy of the database.

records in this fashion can be accomplished remotely with a degree of accuracy far superior to that of parsing a flat file. Metadata information need not be duplicated because both the geospatial data sets located with the data-providers and the GCMD search-engines use the same metadata records contained within the database. When data-providers decide to modify or delete the metadata associated with their geospatial data sets, then the change is reflected in the GCMD database through the modification or removal of metadata records. Also, the GCMD search-engine is capable of querying on any of the DIF metadata fields, which gives it the ability to narrow the search criteria on a very large number of records.

The Global Change Master Directory (GCMD) was originally developed to facilitate interactions among science data cataloguing systems. The success of the system is in its fulfilment of its primary goal of providing science users with the ability to find and view information about science data regardless of which system contains the metadata. However, it uses only a single, relatively simple metadata standard to represent available data, which ultimately limits the representation of more complex forms of geospatial data. The metadata standard dictates the overall design of the system by identifying the sort functionality that can be developed to aid users of the system in storing and retrieving information that is used to reference Earth science data.

The systems examined in this section are implemented in various ways to facilitate the access of on-line geographic information. Common to all these systems is their dependence on metadata for use in describing geospatial data sets, cataloguing diverse pieces of information within system directories and defining search criteria for search-engines. The Alexandria Project (NCGIA, 1995) is a good example of the importance of standardised metadata in the construction of a distributed digital library that provides access to a wide variety of geographic information. However, the type of metadata used is different from system to system. Some systems use standardised metadata while others develop customised attribute/value pairs, which is particular to a specific type of geospatial data. Also, the methods used to store this metadata vary between systems. Some systems rely on flat files to store metadata while others use a database. These system disparities results in considerable barriers

to interoperability between various on-line geospatial data dissemination sites that affects cross-platform searching and processing of on-line geospatial data.

2.3 Metadata parsing programs

The purpose of this section is to discuss contemporary Web-based metadata parsing programs by critiquing how these tools are being used and in what context they facilitate the access and distribution of metadata. The previous section indicates that some on-line metadata cataloguing sites use parsing programs as an effective technique to maintain the currency of their databases. However, with respect to metadata parsing programs, there are appropriate as well as inappropriate situations where these tools can be used. Because of the complexities inherent in geospatial data and the excessive length it takes to describe them using current metadata standards, it is not surprising that the performance of many parsing programs is rather poor for parsing and validating metadata files over the Web (Dodd, 1982). In fact, the performance can be so poor for such tools that it makes their frequent use impractical for Internet applications. Effective use of on-line parsing programs occurs in situations where metadata is initially added to a database in the form of parsing information from metadata files either for entry into the rows, tables and columns of a RDBMS or for validation purposes when uploading such files to a remote server. Ineffective use of on-line parsing programs occurs while performing numerous minor updates of metadata records or files that are already residing within the on-line database. Examples of such are introduced in section 2.4 below.

Among the first programs to effectively validate standardised metadata is called the Metadata Parser (Schweitzer, 1996b), or *mp* for short. The *mp* program is a good example of a utility capable of operating on individual metadata files that are located either locally or over the Web. It validates hierarchically indented text metadata against the 1998 version of the CSDGM standard and converts it to standardised text, HTML, SGML, or DIF forms. The program produces an error report which cites any inconsistencies in metadata text that do not follow the production rules and element

values specified in the CGSDM standard. An extension file, supplied by the user, allows the addition of local extensions to the metadata standard. This changes the output of the parsing program to reflect the requirements of each organisation producing its own type of geospatial data. The *mp* metadata validation program was first developed as a stand-alone utility and was later extended to operate over the Web. Such an extension of its functionality demonstrates that with minimal modification, these types of programs can be converted to operate in a distributed networked environment.

Many of these metadata parsing and validation programs require standardised metadata to be in a particular format within the file for the successful completion of the entire operation (Olsen, 1996). It is not always possible for such programs to predict the format of metadata, particularly with regard to the rather loose definition of conformance associated with CSDGM metadata files. For example, in order for *mp* to operate successfully, the metadata must be hierarchically indented to indicate parent/child relationships between sections, sub-sections and element values within the metadata file. Such a hierarchy is not always present in each metadata file and in many cases the sections, sub-sections and metadata elements are numerically classified. For these reasons, Schweitzer (1996c) wrote *cns* (an acronym for Chew-n-Spit) as a pre-processor for the *mp*. The program takes poorly formatted metadata and an optional list of element aliases and outputs a file containing properly formatted metadata that can be read into *mp* as well as commented output on the text it could not identify. Therefore, additional software is often designed to address limitations of parsing programs that seriously impede their ability to process metadata files with wide-ranging formats.

There is a class of metadata creation tools constructed in the Arc Macro Language (AML) for use within the Arc/Info GIS. These utilities possess the significant ability to extract and/or validate CSDGM metadata from coverages. These AML metadata creation tools can not only operate within a standalone GIS package but can also be used over the Web. The earliest example of such tools is DOCUMENT.AML[9] (Negri, 1996) to extract metadata information such as point, vector or attribute

[9] Subsequent versions of DOCUMENT.AML were developed on UNIX by the USGS and the US Environmental Protection Agency for eventual incorporation into ESRI's version 7 of the Arc/Info GIS.

objects automatically from an Arc/Info coverage. Other tools followed such as BLMDOC.AML (Katz, 1996), Data Dictionary (Daumiller, 1996) and FGDCMETA.AML (Nelson, 1997) providing improved conformance to the CSDGM standard, enhanced functionality and greater ease of use. At present, these metadata creation tools have not been implemented on the Web. But with a minimal amount of effort, both metadata extraction and validation can be performed on-line through the upload and processing of Arc/Info coverages as export files and the return of CSDGM metadata or information regarding metadata compliance.

Therefore, some commercial and freeware metadata parsing utilities are available to software users and developers. However, these utilities are frequently limited with respect to the metadata standards they utilise and that they are standalone applications. Such applications must be downloaded and installed alongside software that already exists on the system. This results in problems associated with system compatibility and periodic upgrades. Nevertheless, it is now possible to extend these metadata-parsing programs to operate over the Web providing metadata validation services to large numbers of on-line users. The next section examines how such parsing programs can be extended to assist in the on-line storage of metadata.

2.4 Metadata creation tools

Among the most daunting problems in implementing and operating an on-line metadata database is maintaining each metadata file or record. As the database grows, the number of files or records becomes far too many to be individually maintained by the site administrator. For such systems it is far easier to allow the creators of the metadata (data-providers) to maintain their own on-line metadata. Hence, there is a need for metadata editing tools capable of accessing remote servers over the Web. The following section is divided into two parts. The first part examines metadata editing programs that consists essentially of standalone software packages that locally store metadata within text or binary document files. The

second part looks at on-line metadata entry and modification utilities that allows data-providers to submit standardised metadata to one or more remotely located servers.

2.4.1 Metadata editing programs

The types of metadata creation tools introduced in this section are stand-alone utilities acting as editors for standardised metadata. In all cases, these editors act on a single document much like a word processing program allowing the user to periodically edit and save changes while subsequently building an entire metadata record. As is the case with editing programs, the process of creating a metadata file in this fashion is oriented toward the user through the use of graphical user-interfaces consisting of windowpanes, display boxes, pull-down menus and tool buttons. Many of these programs have been within the public domain as freeware for some time and therefore warrant a brief examination in order to build a proper perspective for the on-line metadata editing utilities discussed in the following sub-section.

The following section examines these programs in terms of their functionality and ability to operate over the Web. The programs use ASCII text files as the primary medium of metadata storage. Even where a database is used, the software is essentially a stand-alone package with little or no integration among larger Web-based data dissemination systems. Despite this limitation, programs such as xtme (Schweitzer, 1998), Corpsmet95 (Corpsmet95, 1997) and MetaMaker (Hansen, 1997) do provide important insight into the sorts of functionality that can be incorporated into comparable on-line metadata entry and modification programs. Table 2.1 below summarises the functionality of these metadata editors:

Table 2.1 Summary of some metadata editors

	Functionality	Notable developments
xtme	-operates in UNIX X Windows -top window showing the hierarchically arranged metadata element tree of a current file -middle window edits the portion of the tree that is selected -indication of precedence of a metadata element (mandatory, optional or repeatable) -bottom window indicates status -metadata values can be deleted from large portions of the tree -prune function that removes element names with no values	-editor can be extended to use local extensions to the metadata standard -comprehensive help menu defining all the metadata elements -easy-to-use interface that allows select, cut and paste of elements or sections of the metadata tree
Corpsmet95	-left-hand pane for displaying the metadata tree of the record -right-hand pane for entering metadata -files are generated through a 'build' command once identification and metadata reference information is entered	-GUI interface that is intuitive -outputted metadata is hierarchically indented -increasing complexity of metadata tree by expanding section or element names to reveal further branches -use of different icons for precedence values as well as complete or partially complement metadata elements -completion indicator is propagated up the metadata tree (i.e. partially complete child elements reflect as partially complete parents element. - described complex polygon bounding rings and exclusion rings
MetaMaker	-supports the CSDGM and National Biological Information Infrastructure (NBII) standard -database query, report and import/export facilities available	-uses the MS Access RDBMS to store metadata -the Access database is hard coded into the utility

Additional software must be downloaded and installed on the user's system in order to upgrade the functionality of standalone metadata creation utilities. This is unlike metadata creation tools that operate over the Web where upgrading functionality is accomplished in a centralised fashion through the site administrator or developer. For example, none of the metadata creation tools examined above possesses a means to validate the metadata before it is stored within a file or database. Both *xtme* and Corpsmet95 utilities do not verify that the user adheres to the CSDGM production rules and therefore, external metadata verification programs are required such as *mp*

discussed in the previous section. Once distributed, these programs must be continually updated, as new features become available such as a built-in metadata validation utility. In contrast, with on-line metadata creation utilities, the functionality can be dispersed over a number of servers that are maintained by different service-providers. Operations such as metadata validation, database entry, file upload and editing software can be brought together over the Web to form a single cohesive system for creating CSDGM metadata files or database records. These services are all maintained and enhanced over time by the respective service-providers.

However, these early metadata creation tools possess a number of key developments that may be incorporated in comparable on-line utilities. All of the programs have vivid and intuitive graphical user-interfaces that provide multiple windows to map an entire metadata tree (or document), view any metadata section and edit any metadata element. These versatile GUI interfaces provide added usability through animated icons, cut and paste operations, help menus and interactive dialogue. This sort of functionality is difficult to replicate for on-line applications unless one was to use development tools such as Java. However, the performance problems associated with using Java software on the Web preclude it at present as a viable approach for developing potentially very large Web applications such as metadata creation tools.

2.4.2 On-line metadata entry and modification programs

One of the principal issues of concern here is not so much the nature of the program itself but the choice of suitable storage mechanisms into which metadata can be entered and within which it can be subsequently modified. A primary consideration for developers of on-line data dissemination sites is the incorporation of on-line metadata editing tools into the overall system. This section contrasts the various methods currently in use to enter, update, query and delete metadata on-line by pointing out their deficiencies and suggesting possible methods of improvement.

The selection of a storage mechanism for use in on-line metadata entry and modification programs is best determined by examining where these utilities fit into

the overall function of the Web site. These utilities operate from a centralised Web server using either standard text files or a RDBMS as the primary method of storing metadata. An organisation's Web site may possess certain utilities that would benefit from the storage capacity and processing potential of a local RDBMS. In situations where portions of the database need to be queried and the information either sent over the Web or fed into other on-line utilities, then an RDBMS would be the logical choice to store metadata. Sites that are simply required to catalogue metadata might find a standard text file as the best means of metadata storage within a centralised repository such as a Clearinghouse. Other on-line software can be developed separately, such as search-engines for querying individual metadata files.

The maintenance and development costs for storing metadata in a centralised location are considerable. If any of the metadata records need to be updated or removed, then the systems administrator is the only one with adequate access to exact any necessary changes. Allowing users access in maintaining their own metadata files on the system greatly simplifies the problems that arise with updating and maintaining a database. Site developers of geospatial data dissemination systems must decide the degree of database access given to the users as well as the form this access manifests itself. For example, if users are allowed access to their own metadata files what sort of software should be designed to assist in the on-line editing process? What sorts of checks are there to ensure the quality of the metadata placed onto the system conforms to standard? Does the database have adequate security to ensure users view only their own metadata records and not those of others? On-line software associated with geospatial data dissemination systems (search-engines, data processing and GIS analysis programs) depend on the quality of metadata contained in the database. These issues are further addressed in the three examples of on-line metadata entry and modification programs examined below.

2.4.2.1 BIC Metadata Form

The BIC Metadata Form (1997) is a metadata entry and modification program that creates standardised metadata files. The software is used by the Texas/Mexico Borderlands Information Centre (BIC) to create metadata for their geospatial data sets and to catalogue this information onto a federated network of databases known

as the National Geospatial Data Clearinghouse (NGDC)[10]. This on-line metadata editor uses only flat files to store metadata. Therefore, the information is only used for cataloguing (i.e. a search-engine) and viewing purposes. It is not accessible to any other on-line utilities such as data-conversion or other geospatial data processing programs.

The BIC Metadata Form operates from a Web server by using a collection of HTML forms and CGI scripts for submitting metadata to a Clearinghouse. The user-interface consists of a single HTML form divided into the sections and sub-sections consistent with the CSDGM standard. The form is considerably long and comprehensive, with a large collection of standardised metadata elements. It can be completed through a variety of HTML form attributes such as check-boxes, lists, text-entry and text-area boxes. However, the amount of detail provided for the form and its excessive length (Figure 2.4) is perhaps its primary impediment for its widespread use.

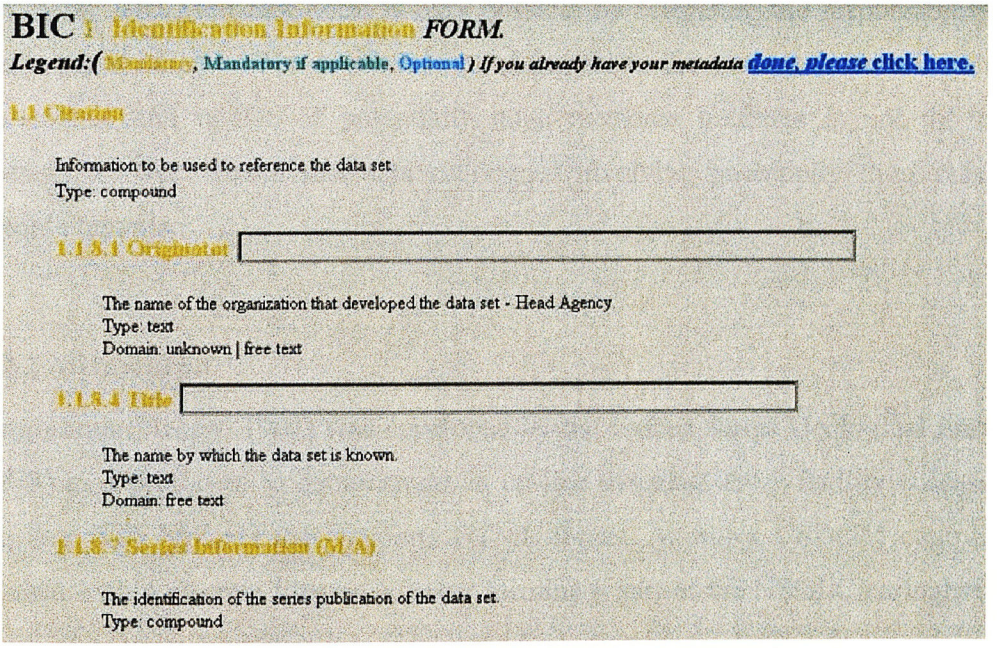


Figure 2.4 A small portion of the Web interface for the BIC Metadata Form showing the colour code scheme used to label metadata elements as being mandatory, mandatory if applicable or optional. The description of metadata elements and HTML form elements are a common feature of such metadata entry utilities.

[10] Or National Spatial Data Infrastructure (NSDI) Clearinghouse.

EDINBURGH UNIV. EX BIBL.

The above figure only shows a sample portion of the form used in this utility; it actually extends for many pages making it far too cumbersome to complete it in its entirety.

The BIC Metadata form does not provide adequate facilities allowing data-providers access to their on-line metadata files. This makes it difficult to maintain very large directories of metadata files because the site administrator must perform retrieval and modification of each individual flat file. The data is processed by a CGI script on the server and placed in a text file, which is then run though the *mp* metadata validation utility and added to the National Geospatial Data Clearinghouse (see sub-section 2.2.2 of this chapter). This process provides no facilities for the user to modify their metadata once it has been submitted to the Web server. As is the case with software developers, geospatial data-providers constantly update and alter their data sets and the metadata that describes these data sets must also be changed accordingly. Updating or removing metadata files from a Clearinghouse using tools such as the BIC Metadata Form is only possible by relegating the responsibility to the site administrator. The centralisation of control to the site administrator for the maintenance and update of potentially huge metadata holdings is not the best approach for Web based metadata catalogues involving potentially thousands of metadata records.

2.4.2.2 Metamorph

Metamorph (Nebert, 1994) was developed by the United States Geological Survey (USGS) as a prototype to demonstrate an on-line metadata entry facility accessible from any Web browser that supports HTML forms. It stores CSDGM compliant metadata within discreet files on a server forming a part of the NSDI Clearinghouse. The utility is unique because it is among the first attempts to allow data providers the ability to construct a metadata record by dividing the task into distinct parts. Therefore, a large metadata record can be completed a section at a time as opposed to filling-out a single (often lengthy) metadata form.

The utility is composed of a series of HTML forms that correspond to the sections of the CSDGM standard and a number of C and PERL programs that are compiled

on the server and used as CGI scripts. The use of multiple forms is a vast improvement over the single and excessively lengthy form associated with the BIC Metadata Form. The user simply completes a particular section of the CSDGM standard and it is conveniently stored on the system within its own file. Therefore, each metadata section can be completed independently and not necessarily at the same time.

Allowing data-providers to have a greater degree of access to the database in order to update their own on-line metadata files frees the site administrator from the tedious task of maintaining potentially numerous files on the server. The final version of Metamorph allows data-providers to modify their previously submitted on-line metadata files. Allowing data-providers to change previously stored metadata is an important functionality for any on-line cataloguing system (Figure, 2.5).

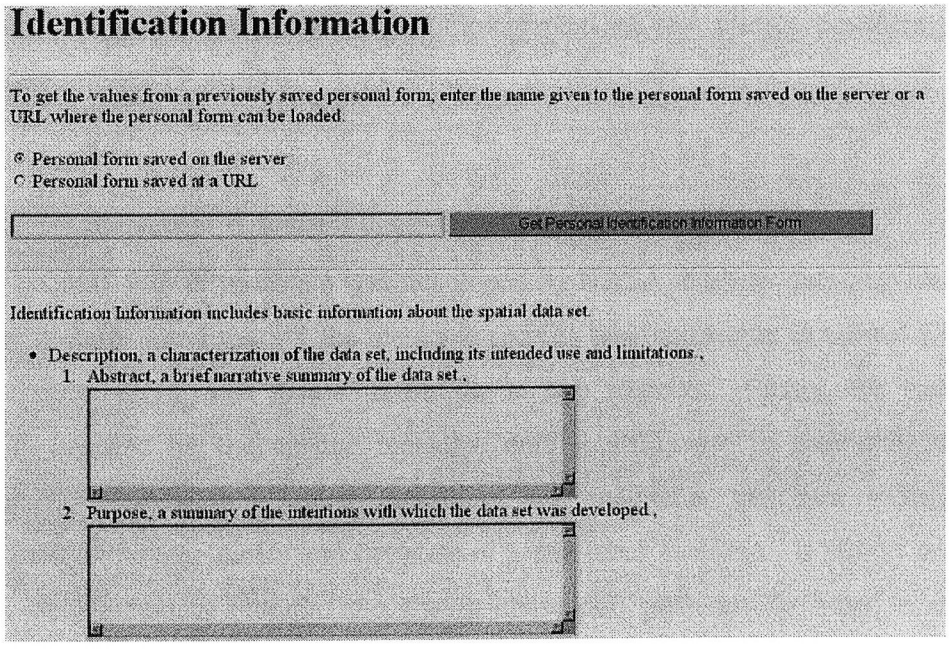


Figure 2.5 A portion of the *Identification Information* form for the Metamorph on-line metadata entry utility. It shows the form-retrieval service that can retrieve forms located on a local server or remotely via a URL. The lower section consists of two free-form text fields in which to enter, in this case, summary and purpose information.

Metamorph accomplishes this task by parsing the previously stored metadata text file and sending the element names and their values back to the user through a CGI script which produces a HTML forms page containing the stored metadata. Parsing text files is an extremely slow process when performed on-line and it explains why the utility experiences problems with slow response times between client and server. The experience obtained from this utility further strengthens the case for using an on-line RDBMS to create, query, update and delete metadata records.

An additional problem is that metamorph does not possess adequate security mechanisms to ensure that only the author of a particular metadata file is able to access it on-line. A better situation would be to implement a secure system whereby data-providers are required to associate their metadata files with a unique identification number or username/password prior to viewing the files on their system. Because Metamorph is no longer being developed, such modifications can not be tested. However, the present functionality can be examined since it is freely available from the Web and can be implemented on any system possessing a C compiler and HTTP server.

2.4.2.3 GCMD: WWW metadata entry form

The GCMD WWW metadata was developed by NASA to allow data-providers the ability to upload and modify the metadata records they contribute to Global Change Master Directory. The system is unique in two respects. Firstly, the metadata records handled by the on-line metadata editing utility are in standardised DIF format. Secondly, this utility modifies and maintains metadata information contained within the tables and columns of a relational database. It allows users to insert new metadata records into the database and provides facilities to update existing database records as well as delete old records. Database records are manipulated on-line using SQL commands and are stored in DIF format. Through the process of database loading, the DIF information is segregated and stored into multiple database tables. The use of a RDBMS and a loader program provide the utility with a degree of efficiency and flexibility that is unmatched in other on-line metadata entry and modification programs.

GCMD is the only utility examined thus far that stores metadata using a loader program. The utility reads metadata in the form of a DIF text file and generates SQL statements, which are used to enter the information into the database. The loader initially compares the DIF fields in the text file to a syntax dictionary and then reads and validates the DIF values before creating SQL statements for insertion into the appropriate columns of various database tables (Olsen, 1996). An important feature of the loader utility is its ability to validate the DIF values before inserting them into the database. It should be added that using an interpreter to generate SQL statements is far slower than the compiled SQL code used in the GMS loading utility described in chapter 3. However, interpreters provide a degree of flexibility for the applications developer that can not be achieved with compiled code.

The utility produces forms of variable length depending on the amount of available information. The length of the HTML forms can be varied cleverly in size allowing data-providers to select only the forms where information is available for the DIF metadata record. Figure 2.6 shows the starting page used to register DIF metadata (Olsen, 1996).

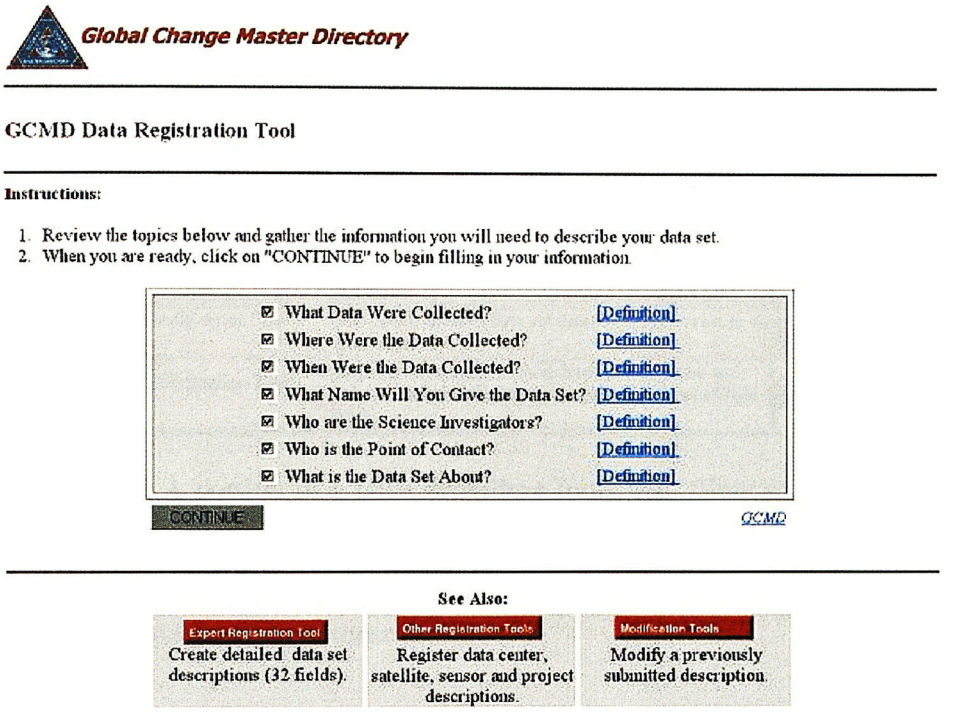


Figure 2.6 The data registration tool that creates a customised HTML form for entering DIF metadata to one or more GCMD servers. The system allows data-providers to construct a form for only available metadata information. Selecting all the checkboxes above produces the entire form for the metadata record containing all of the fields specified in the DIF standard.

The registration tool shown in the figure builds a standard DIF metadata entry form depending on the portions of the form that are selected using the adjacent checkboxes. Pressing the 'CONTINUE' button prompts the CGI script to construct only the selected portions of the standard form. Once completed, the form is sent to the server where the field values are stored in a DIF text file. There is also the option of creating a more detailed DIF metadata record by selecting the 'Expert Registration Tool' button shown in Figure 2.6, which goes to another selection list of checkboxes where the user is able to create a form with up to 32 fields. Furthermore, metadata fields that provide descriptive capabilities not available in the DIF standard may also be added.

GCMD is perhaps the only functional system operating on the Web that allows users to modify their previously submitted metadata records. Because of the relative ease

with which DIF metadata files are parsed and loaded into a database as compared to CSDGM files, GCMD is capable of providing fairly rigorous on-line metadata editing and database entry facilities (Figure, 2.7).

Tool:	Used to Modify:	Link
Data Set Description (DIF) Modification Tools		
DIFWeb DIF Modification Tool (Database)	Data set descriptions that have already been submitted to the GCMD system and loaded into the database.	Modify Record
DIFWeb DIF Modification Tool (Local)	Previously created data set descriptions that reside on your computer.	Modify Record
DIFWeb DIF Modification Tool (Web)	Data set descriptions which have been created and are currently available anywhere on the World Wide Web.	Modify Record

Figure 2.7 A portion of a table providing tools that allow modification of data set records (DIFs). The far-left column specifies tools for modifying DIFs that reside either on the GCMD database, the data-provider's disk space or anywhere else on the Web. The far-right column provides links to these tools.

The on-line modification tools allow the user to edit and re-submit DIF records that have previously been submitted to the GCMD database. An 'Entry ID' is required to associate specific database records with a particular user. Once modified, the DIF text file is sent back to the GCMD system for parsing, validation and database loading. The system also allows users to modify previously created DIF files that reside on their own computers by providing a single HTML text-area field where users can 'copy and paste' their entire DIF file and submit it to the GCMD system for validation. Finally, DIF files can be accessed from anywhere on the Web through the last DIF modification tool of Figure 2.7. The tool provides a single field where the URL address of the DIF file is entered by the user and the file subsequently downloaded by the GCMD system and validated before its entry into the database.

2.5 Interoperability in GIS

The next section examines four different approaches to enhancing the interoperability between different GIS and other types of systems particularly with regard to systems, which operate in a distributed computing environment. It is also important to consider how the WWW can be used as a vehicle for facilitating interoperability between different types of systems (GIS servers, database servers, etc.) that are physically remote from one another yet connected via the Internet. Finally, certain methods are discussed to improve interoperability between on-line systems.

2.5.1 Defining interoperability in GIS

At present, those people who produce digital spatial data largely control its content and structure. This is not surprising since a data set adheres to a particular data model, which is a representation of the developer's abstraction of real world entities and phenomena. The onus lies on the user of geospatial data to determine whether the contents of the data set can be utilised through its implementation on a particular platform or software application (Massam, 1994). The reliance on specific computing platforms and applications is one of the reasons that interoperability remains a major impediment to the exchange of geographic information. With developments in networking technologies there has been a steady improvement in the access of spatial data between heterogeneous databases. Developments in user-interface design, network-browsing utilities and data description/encoding standards have also eased the problems of data flow between disparate systems.

With regard to GIS, interoperability is the degree to which geographic data and services can be accessed in a distributed computing environment (ESRI, 1995). Geographic information and services can be integrated into applications that give transparency to the source and location of the data or any services acting upon the

data. Interoperability in the GIS industry manifests itself in four distinct yet interrelated categories (Sondheim et al., 1999):

1. geodata modelling;
2. cataloguing and metadata;
3. information communities;
4. technology paradigms.

Geodata modelling specifies the meaning of geographic feature and non-geographic objects within geospatial data. This includes the terminology or semantics used for real-world features, the spatial/temporal constructs used to define and represent such features (i.e. points, lines, areas, grids, fields etc.) and the methods or functions incorporated as an integral part of geospatial data. GIS software must interpret spatial data acquired as a non-native format on all three of these geodata-modelling levels in order to use adequately the data.

As mentioned earlier, metadata facilitates the high level description of geospatial data that will consequently enhance or impede interoperability by the nature of its use or misuse within applications. For example, concise definitions of the structure and organisation of geospatial data sets, which consist of different formats for different systems, can be specified in a similar manner thereby facilitating communication between data-providers and users. As previously examined, metadata often forms the basis for the construction of on-line catalogues capable of browsing and searching geographic information from dedicated relational databases or from Web search-engines that reference on-line files.

On-line databases are often federated. They consist of many independently maintained servers that use the same metadata standard to provide a common descriptive terminology throughout the entire network. This enhances interoperability among an information community made up of data-providers and users who share a common understanding of the meaning of data. Information communities form around a number of different mutually agreed standards. These information standards include database schemas, data models, geospatial data standards and metadata standards which must all be internally consistent. No semantic conflicts must exist among the respective user communities. The GIS technologies themselves can be an impediment to interoperability because of their

continual changing technologies and the gradual obsolescence of information standards.

2.5.2 Enhancing interoperability in GIS

The pursuit of interoperability between GIS applications and the wider computer industry in general is given added urgency because of the adoption of distributed computing as a convenient means of exchanging geographic information. The proliferation of GIS and other applications on increasingly smaller and faster hardware platforms facilitates the need to integrate data from multiple sources. There are currently four methods of enhancing interoperability within the GIS industry (ESRI, 1995):

1. direct translators from one GIS format to another;
2. translators that use GIS interchange formats;
3. a client/server model for GIS;
4. a common Applications Programming Interface (API).

The remainder of this section further clarifies these four characteristics that control the degree of interoperability between different types of networked systems.

Before the development of interchange standards for various types of GIS data, vendors used to supply data translators with their products that converted the geospatial data operating within their own systems to other proprietary vendor-specific GIS data formats. The process of direct translation of a geospatial data format from one GIS to another is neither efficient nor accurate. The variations in the accuracy of data translation are considerable because some developers are able to map data models and semantics between two data formats better than other developers. If each GIS vendor produces data translators for all the other proprietary data formats currently in existence, it results in a considerable multiplicity of translators (that very considerably in design and operation) for every possible type of data-conversion.

The development of interchange standards has considerably improved interoperability between different GIS and other software applications. The idea is to translate data to and from various independent, neutral or intermediate formats. For example, SDTS[11] was adopted as an interchange standard by some U.S. federal agencies to translate the geospatial data used within their own application software to and from a mutually convenient data format used throughout the industry (Fegeas et al., 1992). Therefore, each U.S. government organisation need only develop a single conversion utility to translate data in SDTS format that is obtained outside their organisation to a local geospatial data format. Alternatively, locally held geospatial data can be converted back into SDTS format for distribution to other outside organisations and the wider geospatial data-user community. Data translators for interchange standards maintain a higher level of data integrity than direct-translators resulting in a minimal loss of information. Other interchange standards including SAIF, CDF and NTF are described in appendix B. De facto, national and international geospatial transfer standards are of a primitive nature where the user is required to analyse and interpret the data in order to extract useful information (Owen, 1993).

The widespread adoption of the client/server model throughout the computing industry considerably facilitates the interoperability between diverse systems including those that perform GIS operations. Client/server computing is a technology that is revolutionising the computer industry by ushering in a new era of coexistence and openness with flexible system components capable of communicating and exchanging information amongst each other (Orfali et al., 1996a). It is distinct from other forms of distributed computing. Both clients and servers run on different machines with servers providing a service and clients consuming a service. A server can regulate a client's access to shared resources. Clients always request a service and servers passively wait for requests. Both

[11] Spatial Data Transfer Standard (SDTS) is a Federal Information Processing Standard (FIPS 173) for the non-proprietary, vendor neutral distribution of geospatial data and for the exchange of geospatial data between dissimilar digital geographic information systems (National Institute of Standards and Technology, 1992). SDTS supports the exchange of vector GIS features and attributes. Topology is organised into successive 'levels' ranging from the definition of real world entities to the more detailed task of geocoding (see appendix B).

components are loosely coupled and rely on predefined communication protocols to pass messages between each other. Both client and server systems can be scaled horizontally by adding or removing clients as well as vertically by increasing the performance of the server. But perhaps the two characteristics most relevant to this discussion on GIS interoperability are that client/server software is platform independent. Digital data held on servers possesses integrity because of its central location and ease of maintenance. Centrally located data can be accessed by the client using access mechanisms, which are provided by the GIS server regardless of hardware or software platforms. The client/server model not only provides access to on-line databases but also to data-conversion and GIS processing/analysis operations.

There are currently both proprietary and non-proprietary systems that use client/server models to provide GIS functionality over a distributed network. For example, version 7 of Arc/Info contains a Remote Procedure Call (RPC) interface allowing client applications to connect to Arc/Info and invoke any geospatial data processing, analysis or conversion command associated with this product. RPC turns the Arc/Info GIS into a server that is capable of operating even between heterogeneous platforms such as PCs, Macintoshes and UNIX workstations.

RPC initiates a client process (using a synchronous procedure call mechanism) by calling a function on an on-line server. RPC run-time software passes the appropriate parameters to the server in the forms of a message and the server, in turn, receives the request, examines the parameters, calls the procedure and responds to the client with a reply. The entire process or thread must be completed before the client can initiate another request. RPC is a connection-oriented service. Connection-oriented services establish a context of guaranteed delivery of messages between the client and the server. The client and the server interact for the length of the connection before, during and after the data is transferred (Thomas, 1996). In other words, a connection must be established and maintained throughout the duration of an Arc/Info session. Such a process (Figure, 2.8) is adversely subjected to network performance problems such as excessive server-load and low bandwidth, particularly over longer distances.

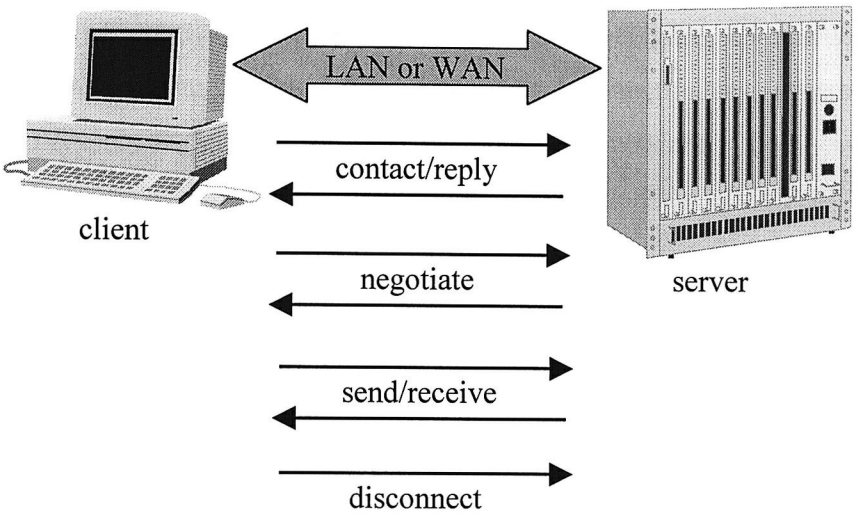


Figure 2.8 A connection-oriented data transmission service using the client/server model operating over departmental Local Area Network (LAN), corporate Wide Area Network (WAN) or Intranets (see Pretty, 1992 for a discussion of these types of networks).

However, RPC suffers from a number of serious drawbacks within a distributed computing environment. For instance, it is not well suited in situations where multiple clients call the same procedure on a server. Separate, non-standardised multi-threaded[12] environments must be implemented to locate servers, to prioritise requests, to check security, to start/stop functions, and to balance network loads. Also, the processors of different computer platforms use a different data structure, which impedes the transfer of data across heterogeneous networked platforms. The Arc/Info RPC is designed to operate in such a computing environment but only within its own family of software products that represents a single-vendor GIS configuration. In this case, a proprietary communication protocol does not address the issues and problems associated with impeding interoperability between different GIS software packages and their associated geospatial data formats.

In contrast to RPC, the Web operates on a much simpler mechanism by transmitting messages between clients and servers using the HTTP protocol. It is an input/output

[12] A process that can handle multiple requests simultaneously. A separate thread in the main program handles each request. This is opposite of the one-process-per-request model used for HTTP. (Yeager and McGrath, 1996)

protocol with a small set of commands designed to retrieve hypertext (constructed using HTML) documents. These documents are referenced throughout the Internet by URLs. HTTP operates on a request/response model (Figure, 2.9 and see section 1.2 of chapter 1). The model is often known as 'connectionless' communication (Thomas, 1996) because each message is treated independently from every other message. In essence, the network, comprising of HTTP on top of TCP/IP[13], can not distinguish between a request and a response message (only the client and server can do that). It simply delivers a message between two machines given a URL address. HTTP is a 'stateless' protocol because the server does not remember previous requests. Therefore, each request requires a separate TCP connection. For example, downloading an HTML document containing five images requires six separate requests to the server, one for the document plus five for each image.

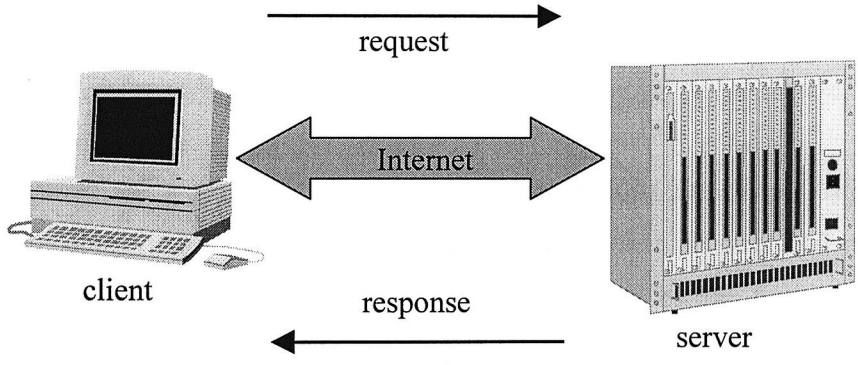


Figure 2.9 A 'connectionless' and 'stateless' data transmission service using the request/response model operating over the Internet.

The advantage of connectionless verses connection-oriented network delivery services is that a connection does not need to be maintained over long distances where low bandwidth and server-load becomes factors when transferring data between clients and servers. Also, the Web's associated protocols are internationally recognised standards used throughout the world to operate the largest computer network in existence. Overall, the simplicity and standardisation of Web technology

[13] Transmission Control Protocol/Internet Protocol

makes it very effective in developing interoperable GIS applications functioning on one or more servers and accessible by a potentially vast number of clients.

HTTP is easy to implement and the stateless nature of the protocol is suited for simple client/server environments, which maximises the performance of servers operating over long distances (Brebner, 1997). However, this is achieved through a decrease in efficiency of the protocol because it must establish a separate TCP connection for each request. Complex operations requiring multiple steps are difficult to execute over the Web because the network is incapable of remembering previous commands. This presents an interesting problem because the ability for the servers to remember client attributes during a connection (or tracking state) is essential for performing complex client/server interactions such as GIS operations over the Web. Techniques to circumvent this problem can be constructed using HTML *hidden* fields and CGI scripts as well as the Java programming language. HTML hidden fields are used throughout the GMS implementation described in chapters 3 and 4.

The development of a common Application Programming Interface (API) is a further step in achieving interoperability among systems that process and analyse geospatial data (ESRI, 1995). A common API is not affected by any future changes made to geospatial data formats. This is often the problem when data standards are successively upgraded causing dependent software applications to upgrade alongside them. Geospatial data can be accessed through a commonly agreed query language similar to SQL. SQL is the standard API for relational database management systems. A common API can support all of the GIS-related data processing and analysis capabilities that are accessible throughout an organisation or over the Web using the client/server model.

The success of API technology and its impact on distributed computing within GIS can only be realised though the development of a single API for the entire GIS industry. A number of GIS vendors have pushed ahead in developing an API for their own systems such as the Spatial Database Engine (SDE™). The system reads and writes binary entity data using software tools developed by ESRI that are accessible within a client/server computing architecture. In order to advance this

approach beyond the realm of specific GIS vendors, a common data model is also required for use in a variety of software applications. A common API means a uniform interface for all GIS servers. This results in more portable applications, which are easier to develop. In addition, a common data model allows data to be exchanged between clients and servers without the need for any conversion. Rationalising both APIs and data models throughout the entire GIS industry requires a considerable amount of co-ordination among GIS vendors, government agencies and academic institutions.

2.5.3 Distributed GIS functionality

Sharing existing software, computer and data resources across a network of computers are vital prerequisites for the development of federated network information systems that store, process and display geographic information. Web technology provides a considerable amount of interoperability between different systems that exist across the network. There are two requirements to consider in the design and implementation of systems that provide GIS functionality over the Web (Abel et al., 1994; Camarata, 1992; Bresnahan et al., 1994).

1. A consistent user-interface, a Web browser or something built specifically for a proprietary software product.
2. A considerable amount of flexibility in assembling the modules of on-line GIS applications to solve inadequately defined or ambiguous problems so often associated with GIS analysis.

The problem of sharing information between diverse components of on-line GIS systems spread over a federated network is one of system integration that enhances interoperability. An organisation can use its existing software or commercially available products to minimise the costs of developing GIS applications over the Web. Systems constructed in this fashion provide a degree of flexibility since components can be assembled from the bottom-up in accordance with any special contingencies required in dealing with unique spatial problems. For example, both the Global Earthquake Mapping and Information Package (GEMIP) and the World-Wide Earthquake Locator (Towers and Gittings, 1995; Towers, 1993) are distributed

systems operating over the Web, which consist of distinct components designed to store, display and process spatial and attribute data related to earthquakes.

GEMIP was developed as a distributed earthquake information database to demonstrate how data management and analytical system components can interact in a distributed Internet environment. The system addresses the quality and currency of data through the update of global seismic events, as they are available over the Internet. The seismic data is obtained from the USGS National Earthquake Information Centre (NEIC) archive located on a server located in Goldon Colorado. The data set is the most extensive record of seismic activity available with records that include the source, date, location, magnitude and intensity of the event. NEIC provides access to recent earthquake information over the Internet through the 'finger' protocol (i.e. finger gldfs.cr.usgs.gov). GEMIP automatically obtains this data through a background process by 'fingering' the NEIC server at 24-hour intervals. In this manner, the local database is kept current with information on earthquakes above 5 on the Richter scale. Once retrieved, the data can be analysed and even displayed to researchers with an easy to use software package.

The World-Wide Earthquake Locator is a good example of distributed system components with distinct functional characteristics being controlled from a central location on the Web. The Locator was developed to provide seismic data on a global scale using the multimedia potential of the Web. It queries the USGS earthquake data archives at NEIC in real time (as opposed to every 24 hours) using the finger command. The data is processed by the Locator interface in Edinburgh and displayed using the PARC Map Viewer (Putz, 1994) at Palo Alto, California. NEIC, the Earthquake Locator and PARC form a truly distributed system of data storage, processing and display. The Map Viewer, NEIC, Earthquake Locator and the user comprise four distinct components that can be potentially placed at different locations over the Web yet still operate as a cohesive system.

2.6 Querying geospatial metadata

So far the discussion has focused on various standards to represent and search on-line metadata, procedures to store and modify it and a number of interoperability issues that concerns the operation of GIS over the Internet. However, within distributed systems, perhaps the most important design aspect is the way the Web can be used to search remote databases. There are a number of different methods typically employed. This discussion is not an exhaustive account of the types of search-engines used by developers of Web sites that provide access to spatial data. However, the systems examined in this section exhibit a range of technology currently available as well as how they are integrated with the services provided by different on-line geospatial data dissemination sites. Of particular importance is the design of the user-interface since it not only controls all other system components but also introduces new user to various system functions.

2.6.1 Digimap interface

The Digimap system was previously discussed in section 2.2.2. The facilities provided by the on-line application to query and display information employ an applet interface that possesses a number of features common in utilities that use certain Java classes to animate the search-engine and network with the server. Beginning with a map of Britain, locations and associated data sets can be searched by keyword; places can be selected and highlighted with the mouse or OS sheet names can be defined by entering bounding co-ordinates. Users can 'zoom in', 'zoom out', 'zoom to a scale', pan, drag out a rectangular search area or select 100KM grid squares. Depending on what the user does, requests are continually sent to the server and resulting maps are continually converted to GIF files for display so as to give the impression the user is navigating through maps of increasing or decreasing scales.

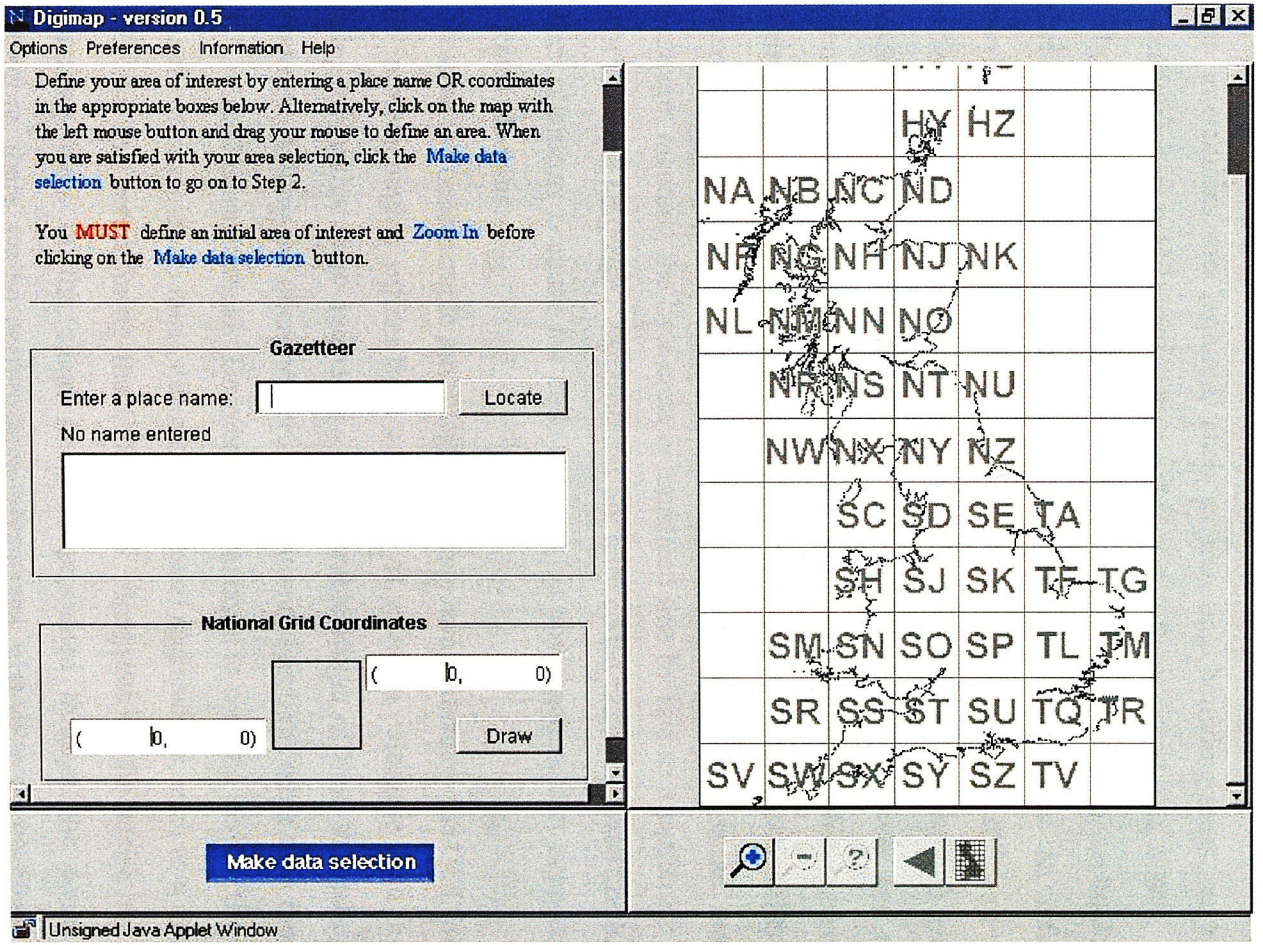


Figure 2.10 The right part of the Digimap user-interface shows a pan and zoom map of Britain developed using Java. The left part of the interface shows text and textarea fields similar to those used for HTML forms to query the metadatabase. However, these forms were also developed in Java using the classes within the Java abstract windowing toolkit (java.awt) package.

During the development of the user-interface, it was obvious that most of the required functionality for the Digimap service could not be implemented using static Web pages containing HTML forms that sent information to CGI scripts. There are many instances where image-maps are the most effective means to deliver information over the Web. However, the interface associated exclusively with HTML forms is rather cumbersome to use in situations where relatively complex user interactions are required, such as selecting bounding-areas on a map and detecting errors using client software (Medyckyj-Scott and Morris, 1998).

The developers of Digimap therefore decided to use Java technology as an alternative to HTML forms/image-maps in order to produce an interface that provides the sort of functionality one would come to expect in software requiring a considerable amount of interaction with the client. Java programs are 'tagged' within HTML pages as applets that are retrieved from the server and run on the client's Web browser when downloaded. The Java language is object oriented whose classes provide all the vivid GUI (Graphical User Interface) functionality described above such as panning, zooming and area selection. Java is an interpretative language, meaning the byte-code is translated line by line to a particular platform's own machine language. Applications are only coded once and need not be ported to other platforms because the code operates on any machine that supports the Java Virtual Machine.

Despite the considerable advantages of using Java to develop the GUI for Digimap, the implementation suffers from a number of setbacks. Because Java interprets its code, it is rather slow and takes a considerable length of time to load-up onto the browser. Asking the client to perform additional functions other than those that are GUI related is impractical. Despite the advantages in creating vivid GUIs with Java, its networking classes must still access backend applications, such as a database or a GIS, via the server using the CGI model. Requests to the server are slow and responses back to the client must be processed through the same interface. Therefore, there are no performance advantages relative to an interface composed entirely of HTML forms. Java is still in the process of development and is not compatible with all computer platforms.

2.6.2 ScotLIS WWW interface

The Scotland Land Information System, a prototype system designed by Harvey (1996), is an effective demonstration of using HTML forms, image maps and CGI scripts to provide a quick and effective database access of land and property information of Scotland. The interface designed for ScotLIS resides entirely on the server. The use of a simple Web browser to operate the interface provides an important distinction between systems that enhance the functionality of the browser through Java applets (e.g. Digimap), plug-ins (e.g. SVF or INSO CGM) or specialised Web clients (e.g. ArcExplorer). Properly distributing the functionality between the client and the server is an important consideration with Web applications that query, process and display geospatial information. Too often the developer fails to match the proposed functionality of the Web application with the correct type of technology and in the process creates something that performs far worse than software developed with more conventional tools.

The first on-line geospatial data dissemination sites used image-maps to retrieve geographic information from a server. Today, this HTML add-on still remains a vital component of such systems. The ScotLIS starting page (Figure, 2.11) consists of an image-map of Scotland partitioned into county areas (Harvey, 1996).

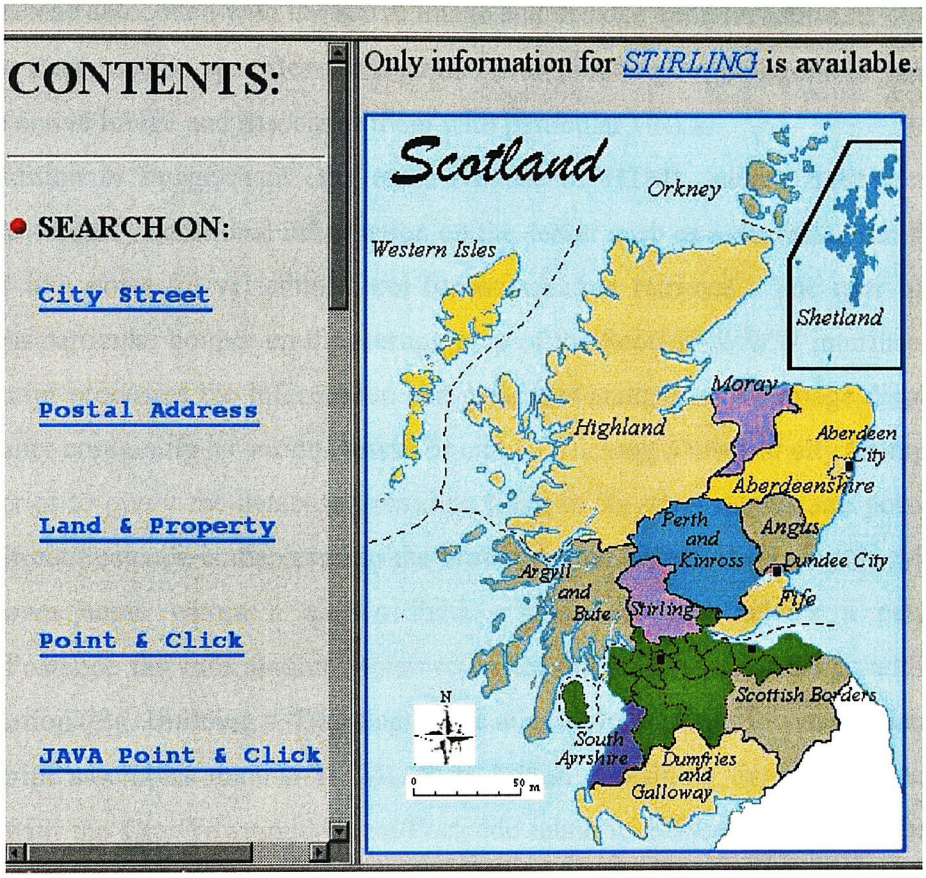


Figure 2.11 The ScotLIS user-interface displaying the applications menu in a separate frame to the left and the main display area to the right.

Image-map is actually a C-language CGI-compliant program for handling active image data, which can be used on any HTTP server supporting the CGI specification (Graham, 1996). Activating an image within a Web browser allows users to access various services located on the server depending on the location of where the mouse pointer is clicked on the image. The services can be diverse including accessing other HTML documents, querying database records, viewing other graphical images or downloading executable files. Active images are much like hypertext links except the request from the client reaches the server in a slightly different way:

- The client measures the co-ordinates of the mouse pointer when the user clicks on the active image and sends this information to the server.

- The gateway program on the server interprets the co-ordinates by referring to a database associated with the active image and decides upon an action.
- The database file specifies the regions (circle, rectangles, polygons or point) of the active image and associates them with particular URLs.

In addition to image-maps, the extensive use of HTML *submit* buttons within ScotLIS to query additional information on the server such as a database, flat files or images is a powerful yet simple tool for information retrieval. The user initially selects a particular county on the starting map of the ScotLIS WWW interface. The CGI script processes the information and generates another active image displaying the county containing its various towns or cities. Clicking a town or city prompts the CGI script to query the database providing a list of its street names and postcodes. Each Street/Postcode is displayed to the browser as an HTML forms button with the City/Town name within a *hidden* field. When the user selects a particular Street/Postcode the information contained in the button is used to query additional information (e.g. Building #, Building Name and/or Organisation) from the database. The script assembles another HTML form button containing the new information along with the City/Town and Street/Postcode name in *hidden* fields. The process can continue extracting greater amounts of detail about land and property information in Scotland. HTML form buttons can contain complex query information but are relatively easy to use and implement.

2.6.3 GCMD search-engine

The Global Change Master Directory is a good example of how a system continually evolves with successive developments in client/server technology in order to provide an interface for accessing the GCMD federated database. A textual interface is among the earliest interfaces constructed for this system. A client in the form of a terminal window allows users to perform a search on the GCMD master directory server as part of a telnet session. The interface provides a series of numbered menus that are selected by entering a number on the command line located at the bottom of the screen. Input is entirely by keyboard and output is only in ASCII format. However, the interface does provide the user with the option of

entering fairly complex SQL queries. The original interface has since been discontinued and replaced with one that operates in a UNIX X-windows environment. An updated textual interface continues to provide users of dumb terminals with the ability to access the master directory (Northcutt and Shein, 1998).

The Web provides GCMD developers a tool to design a search-engine that uses standard HTML forms that are cleverly organised into specific categories of earth science data. The initial version of the WWW interface consists of a series of HTML text fields for each type of data category, all of which are positioned on a single page. If the terminology for a particular category is not clear then the user can press a button beside each field to select a list of valid terms. The Web interface was updated and redesigned to accommodate up to eight search criteria with a possibility to select a maximum of four keywords for each criterion (Figure 2.12). As before, pressing any of the category buttons reveals an HTML scroll bar displaying a choice of valid terms for the category used in the database search. Pressing the *Start Search* button sends the form information to the CGI script, which assembles the database query. The design of this particular search-engine is best suited for databases containing standardised metadata because of its reliance on carefully defined terms for description and categorisation.

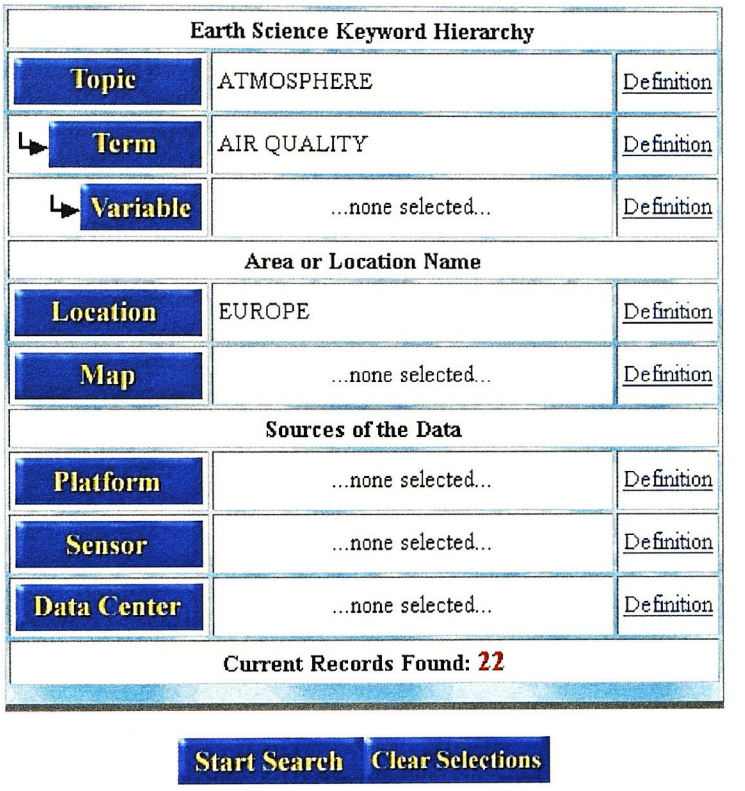


Figure 2.12 The GCMD guided search-interface that assembles database queries which access the on-line master directory. Each button to the left reveals additional form components designed to narrow the search query for a particular criterion.

2.7 Web robots in GIS

As discussed previously, a distributed system would ideally possess the ability for automated checks, updates and retrievals of data located at different remote locations. Automated programs to provide such functionality over the Web are frequently referred to as 'Web Robot'. More generally, Web Robots are a class of programs called *agents*. They consist of software technology capable of performing tasks for their users more efficiently than humanly possible or by simulating human-like actions (Cheong, 1996). Agents can be categorised on the basis of:

- Human-delegated software entities that perform tasks considerably faster than if performed manually,
- Software that facilitates work and co-ordinates tasks among people, machines, and other agents,
- Repositories of knowledge that facilitate teaching and are capable of learning new things,
- Software capable of creativity, emotion and intelligence.

Given a very brief overview of a considerable large area of study, this section is only concerned with the first category by describing the most commonly occurring types of agents currently in operation over the Web. This category includes terminology such as Web robots, spiders, wanderers or brokers. Such software traverses the WWW by moving from one HTML document to another by referencing the hyperlinks embedded in each Web page. Web robots discover new resources, index cyberspace and search for dead links using HTTP to retrieve documents from servers for parsing and examination purposes.

Some of the more commonly used Web robots such as AltaVista and WebCrawler allow users to submit the URL addresses of a particular Web site. The robot traverses the Web site in the background and indexes the addresses of the documents within its database. The database can then be searched using one or more keywords or strings, which are entered into a text field of an HTML form (i.e. search-engine). These Web robots possess enormous databases accessible via the Web using search-engines that organise information using a number of approaches, which include cataloguing, inverted indexes, latent semantic indexing and knowledge base (DeBra and Post, 1994; Randall and Randall, 1995). In addition, the Web robots are capable of building lists of URL addresses to access while traversing Web documents. This allows the software to operate on a continual basis for indexing the WWW consisting of millions of servers and hundreds of millions of documents.

The technology developed for today's Web robots can be incorporated into software that specifically indexes the hypertext links referencing on-line geospatial data sets, metadata files, images and other related geographic information. The proliferation of geographic information over the last few years is due, in part, to the increased use of

Web software in facilitating the accessibility of such information along with a decrease in the cost of computing hardware such as storage devices and processors. The result is an increasing reliance by GIS users and geospatial data-providers on the Web for browsing, searching and downloading all types of geographic information. The amount of on-line geographic information is already considerable, thereby requiring automated means to find and catalogue only the relevant bits of data contained within numerous servers distributed throughout the world. Web robots and other types of Internet agents can provide the fast and effective functionality to automatically traverse large portions of the Internet for the purposes of indexing geographic information as well as for information discovery. The Web robot introduced in this sub-section simply trawls around for hypertext links of on-line file documents. However, future research may be able to incorporate into geospatial Web robots some of the most impressive features seen in today's more common commercial and research-oriented Web robots.

2.8 Chapter conclusion

This chapter examines a broad range of technologies that are currently implemented in varying degrees within on-line geospatial data dissemination sites. Some of the technologies are fundamental in constructing such systems because they address some key problems and issues that impede the access of geographic information over the Web. The use of metadata standards is one such important consideration for the development of such Web sites due to their ability to tie existing system components together as well as provide a common basis for expanding functionality to other system components or even other Web sites.

Certain storage methods for metadata are better suited than other methods depending on the type of service provided. The method used for on-line metadata storage is another consideration when designing such intertwined systems. There are also a variety of different methods for building and maintaining on-line metadata databases. These must be examined in terms of the degree of database controls and

database access these Web sites bestow to site administrators vis-à-vis geospatial data-providers. Technologies such as metadata parsing programs, standalone metadata creation tools and on-line metadata editing/modification utilities are of considerable importance in constructing metadata databases.

The Web provides a vehicle for facilitating interoperability between GIS servers, database servers and other types of systems that are physically remote from one another yet connected via the Internet. The indexing and cataloguing of metadata and its query using a range of sophisticated search-engines provides the necessary means for geospatial data-users to locate all forms of geographic information over the Web. The deluge of information affects some key aspects of GIS technology that have traditionally suffered as a result of poor interoperability between terminology, data formats, software applications and computing platforms. There are different approaches to deal with the many facets of interoperability as it manifests itself within the GIS industry. However, the WWW is the approach of particular relevance to this research.

CHAPTER THREE

METADATA ENTRY AND STORAGE

This research has produced a metadata database and a suite of database utilities and on-line software collectively called the Geospatial Metadata Server (GMS). These programs whilst having similarities in design with existing systems are all newly written. The entire system consists of six components. Three of these components, (metadata database, database entry utility and metadata editing utility) are described in separate sections of this chapter. They consist of a database schema and associated utilities designed to enter, store and maintain on-line metadata. The metadata database is implemented within a RDBMS. The database schema uses the CSDGM standard to store metadata. The standard provides geospatial data-providers and data-users a common format of descriptive terminology. Due to the excessive length of metadata records, GMS is equipped with a parsing algorithm and database entry utility to insert delimited metadata text files into the database. It also provides a metadata editing utility, which is divided into a series of HTML forms each corresponding to a database table. The utility cleverly allows data-providers to maintain their personal metadata records over the Web.

The primary method of evaluation of these three GMS components is through on-line use followed by e-mail feedback. To facilitate this process, the parsing/database entry and metadata editing utilities are advertised within comp.infosystems.gis Usenet newsgroup as tools that directly access an on-line relational database that stores standardised CSDGM metadata files. Select individuals the author deems as important contributors to the human evaluation of these GMS components can be contacted directly and encouraged to use these on-line utilities. To make the evaluation process easier, a detailed on-line explanation of how these utilities operate is provided. A few sample CSDGM metadata files can be entered into the metadata database via the parsing/database entry utility and used to provide an on-line

demonstration of the metadata editing tools. Users can either operate the editing utility by the manipulation metadata attribute-values pairs from the sample metadata records or they can insert their own metadata into the database using the parsing/database entry utility. The author responds to any problems, comments or insight received via email from the on-line users and the results of the transactions are documented in the evaluation and discussion (Chapter 5) part of this thesis.

3.1 The GMS metadatabase

The metadata database forms the core component of GMS. The database schema is designed using a portion of the metadata element and section names contained within the CSDGM standard. It is implemented within the ORACLE7™ RDBMS (Abbey, and Corey, 1995; Gittings and Healey, 1989). All other GMS system components either access the database directly, via tools that maintain and query on-line information, or indirectly, using data processing and cataloguing tools. Therefore, a detailed description of the design and implementation of the GMS metadatabase is required to achieve a comprehensive understanding of the entire system. This is supported by commentary on using metadata standards as way to distribute geospatial data over the Internet.

3.1.1 Metadata standards

The CSDGM standard used for the GMS metadatabase is much broader in its descriptive capabilities for geospatial data than the Directory metadata Interchange Format (DIF). For CSDGM compliant metadata to be of any use in Internet applications, it must be able to be parsed via algorithmic means for entry into a software system and it must be interoperable with other Internet based systems that use the same type of metadata. The CSDGM is a content standard capable of providing a comprehensive description of most types of geospatial data. As described in section 2.1.1.1, the standard consists of a very large number of

descriptive elements divided into a meticulously designed hierarchical structure comprising (at the time) the most comprehensive metadata standard for the description of geospatial data. However, because of its emphasis on data description, the standard relegates structure and layout of the metadata file to its creator (Schweitzer, 1996a). This allows geospatial data-providers a rather broad definition of conformance to the standard because the terminology used to describe a data set may be consistent with other data-providers and organisations but its structure and layout varies considerably. This fact precludes the usefulness of CSDGM metadata over the Internet because associated software must be able to index, search and retrieve metadata documents on-line. Using the information contained within compliant CSDGM metadata files for purely visual purposes (as files to be download in their entirety and viewed) seems a rather narrow utilisation of metadata in general. Through the parsing of standardised metadata and its exchange between various system components, as is the case with the GMS implementation, metadata in general performs a central role within Web applications designed to access on-line geographic information.

The GMS implementation is a far more comprehensive attempt at using a very large and complex metadata standard as CSDGM for Web-based applications. The GMS metadatabase, is not a full implementation of the CSDGM standard. However, for testing purposes the number of database tables and metadata elements used is more than adequate to demonstrate system functionality. Because the metadata is standardised, it is predictable in form and content for those who access Web-based geospatial data dissemination sites. The predictability ensures the metadata can be parsed and made available to other systems and applications located throughout the Web.

3.1.2 Software design

Because of the complexity of the CSDGM standard and the time constraints placed upon this project it was decided to implement only some of the sections and elements contained in the standard. This sub-section introduces the semantic data model used

in designing the metadatabase. The analysis method used is a system-modelling notation known as entity-relationship modelling. Such a notation specifies the logical data structure as a collection of tables composed of entities, attributes and relationships. Entities are objects or concepts that can be uniquely identified, are capable of an independent existence and are designated with their own table within the entity-relationship model. Attributes are properties of an entity forming the columns of an associated database table. Entities or tables possess certain relationships between themselves which can either be a direct relationship between two entities, *1:1 relationship*; or a relationship where one entity is related to a number of other entities, *1:many relationship*; or the case where entities inter-relate with each other, *many:many relationship*. For further reading on entity-relationship modelling the reader is referred to Howe (1989), Date (1986) or Chen (1976) for a more in-depth description on the subject.

The entity-relationship model for the GMS metadatabase comprises of fourteen tables, seven look-up tables and a plethora of relationships. The model is shown in Figure 3.1.

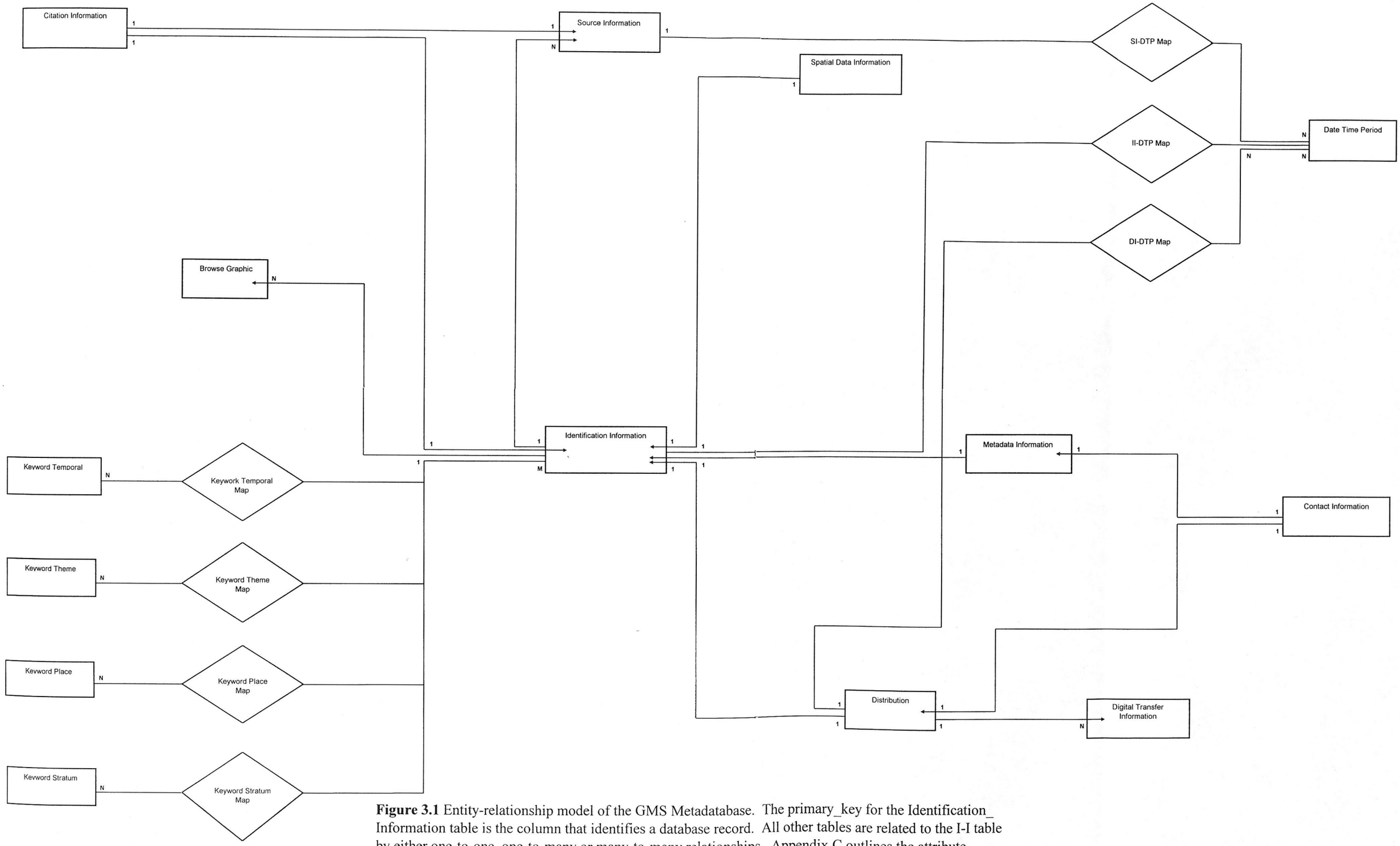


Figure 3.1 Entity-relationship model of the GMS Metadatabase. The primary_key for the Identification_ Information table is the column that identifies a database record. All other tables are related to the I-I table by either one-to-one, one-to-many or many-to-many relationships. Appendix C outlines the attribute assignments for each table.

The entities selected to model the metadatabase correspond to some of the sections outlined in the CSDGM standard. In order to explain the design of the GMS metadatabase it is important to include the same terminology present in the CSDGM standard. The rational behind including certain tables and columns in the database schema and the relationships between each table can be discussed by referring to the organisation of CSDGM sections, compound elements and elements. As mentioned in chapter 2, the standard is organised in a hierarchy of data elements and compound elements. The starting point is *Metadata* (section 0), shown below. *Metadata* is a compound element composed of other compound elements representing different concepts about the data set. For example, *Identification_Information* (another compound element) is designated as section 1, *Data_Quality_Information* as section 2 and so on. These top-level sections are further divided into smaller sections labelled as 1.1, 1.2, 2.1, 2.1.1, 2.1.2, 2.1.2.1 etc. that consist of more compound elements and elements. The following notation is used to represent these relationships:

Metadata =

Identification_Information + *0{Data Quality_Information}1* + *0{Spatial_ Data_Organization_Information}1* + *0{Spatial_Reference_Information}1* + *0{Entity_ and_Attribute_Information}1* + *0{Distribution_Information}1* + *Metadata_Reference _Information*

This terminology is compiled in a series of 'production rules' between different elements and compound elements specified in the standard. Much like mathematical equations, production rules have their 'identifiers' on the left side and their 'expressions' on the right thereby using substitution of data elements to explain higher-level concepts. Productions rules are controlled by the following operations in addition to the one explained above:

= represents a substitution of the expression with the identifier,

+ represents an 'and' operator,

[|] is like the C language exclusive or operator where a single term is selected from the list enclosed in square brackets,

m{}n represents an iteration where the term(s) in the brackets are repeated from m to n times,

() indicate optional term(s).

The production rule, 0{ term }1, which designates a metadata characteristic as 'mandatory if applicable' if in the event the data set displays a particular characteristic then such information must be included as part of the data set's metadata.

3.1.3 On-line metadata and relational databases

By adhering to a common metadata standard as opposed to ad-hoc methods of data description, different on-line geospatial data dissemination sites are capable of integrating their search and query functionality. This is particularly true for federated databases where interoperability between the disparate databases is achieved through standardised metadata. This allows each on-line software system to read the metadata, identify its elements and function in accordance to system design.

Integration of on-line database systems involves the access of information that exists on different systems using different storage mechanisms and maintenance facilities. The information can be combined from multiple sources (data selection, update or entry) and used within applications capable of interpreting the information to perform some useful task or operation. A schema designed at the conceptual level is used to network and integrate multiple heterogeneous databases consisting of relatively diverse systems (Orlowska et al., 1992). Integration at the conceptual level within databases entails a consistency in names, types, values, formats, descriptions and data-structures. Such databases also require additional software known as trigger[1] mechanisms that ensure the compliance of any constraints or rules placed on the data as it is being entered or queried from the database.

[1] Database triggers are stored blocks of a query language that are associated with particular tables and are executed when a specified SQL statement is issued against the table (ORACLE7 Server, 1992).

A number of different methods used to store on-line metadata are currently implemented on the Web. These provide varying degrees of control to site developers in deciding metadata form, terminology and definition. The Global Change Master Directory is made up of a number of heterogeneous databases forming a unified view of an entire collection of metadata files (Olsen, 1996). Each database accepts either SQL queries or special query commands such as GET which retrieves either formatted sections of a DIF or an entire DIF file. The DIF metadata records are stored and maintained within an RDBMS. This is similar to the GMS metadatabase implementation where a portion of the metadata standard is implemented as tables and columns in a relational database.

It would be possible in future developments of the GMS metadatabase to expand the database schema so as to make it more representative of the actual standard. A larger metadatabase would include expanding existing tables with CSDGM compliant elements and creating new tables by adding additional sections. In particular, the metadatabase can be made more representative of the actual standard by modelling the three CSDGM sections, *Spatial_Data_Organization_Information, Spatial_Reference_Information, Entity_and_Attribute_Information,* as separate entities (i.e. tables) as opposed to using a single table called *Spatial_Data_Information.* Since the database forms the basis on which all subsequent software is constructed, any changes in the database require modifications to various GMS components. However, a fully operational on-line metadatabase using a recognised national or international metadata standard is an important prerequisite for the widespread use of geographic information over the Internet.

Metadata that can be parsed forms an integral part of the system by providing interoperability between the various system components of on-line geospatial data dissemination sites. The Global Change Master Directory is one of the few examples of a system that implements an entire metadata standard within an RDBMS. Other on-line systems store their metadata as flat files (e.g. NGDC see Plewe, 1994), which are distributed throughout a federated network of servers. The only CSDGM metadata elements committed to a relational database are those deemed useful for a Web search-engines such as certain elements from the *Citation Information,*

Keyword and *Identification Information* sections of the standard. Alternatively, some systems use ad-hoc methods of data description (GENIE, DIGIMAP see Medyckyj-Scott et al., 1996 and Medyckyj-Scott and Morris, 1998, respectively), which are then stored as flat files on various federated networked servers and accessed via custom built search-engines. Despite the importance of metadata within geospatial data dissemination sites, none of these Web applications use a geospatial metadata standard as a basis in cataloguing data sets within a relational database.

GMS uses a RDBMS to store standardised metadata capable of being parsed and passed to other software components. This facilitates overall system interoperability. The database provides a remarkable degree of flexibility for data-providers and site-administrators to select, enter, update and delete metadata information from the tables and columns of the RDBMS and send it to other system components that are located either on the local server or the Web. The design of the metadatabase allows the site developers to construct queries capable of searching any section or element of a particular metadata record. In this fashion, specific amounts of information can be made available in segments so as not to overload the user with too much information. The information can also be assembled and viewed in any shape or form such as flat files and HTML documents or sent over the Internet within distributed objects.

3.1.4 GMS metadatabase implementation

A detailed description in implementing the GMS metadata database is presented in this sub-section. This includes a summary of the top-level sections for the CSDGM standard and their relevance towards the design of the database tables and the relationships between tables. For instance, the *Identification_Information* (I-I) table is mandatory by forming the core of any CSDGM metadata document. This is the primary table in the metadatabase upon which all other tables are related and all metadata records are indexed. The elements that comprise this table are elaborated further on in this sub-section. The *Identification_Information* table has a number of

key components, which describe basic information about the data set. The information is summarised using the following production rule:

Identification_Information =
Citation + *Description* + *Time_Period_of_Content* + *Status* + *Spatial_Domain* + *Keywords* + *Access_Constraints* + *Use_Constraints* + *(Point_of_Contact)* + *(1{Browse_Graphic}n)* + *(Data_Set_Credit)* + *(Security_Information)* + *(Native_Data_Set_Environment)* + *(1{Cross_Reference}n)*

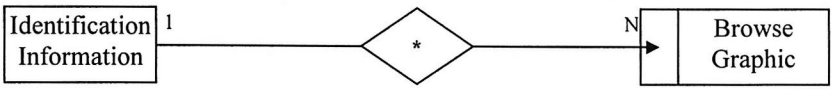
The production rule above shows that the *Identification_Information* section comprises of a number of other sub-sections and elements, some of which are used in the modelling of the metadatabase. In particular, *Citation_Information*, *Browse_Graphic*, *Time_Period_of_Content* and four *Keyword* tables are modelled as separate tables within the metadatabase. They are either linked by one-to-one, one-to-many or many-to-many relationships with the *Identification_Information* table. For example, the *Citation_Information* table is linked not only to the *Identification_Information* table, but also to the *Source_Information* table which is actually a subset of Data Quality Information (a top-level CSDGM section). See Howe (1989) and Oxborrow (1989) for a detailed discussion on the development of a conceptual data model using a top-down approach that identifies entity and relationship types.

The *Browse_Graphic* table stores reference and access information for one or more graphic representations of the geospatial data set. It is part of the identification information section of the CSDGM standard using the following production rule.

Browse_Graphic =
Browse_Graphic_File_Name + *Browse_Graphic_File_Description* + *Browse_Graphic_File_Type*

Since there is the possibility of having more than one graphic file for a particular data set, a one-to-many relationship is implemented between the *Browse_Graphic* table and the I-I table. Membership was deemed obligatory for the *Browse_Graphic* table

because graphic files must be associated with a particular data set. Diagrammatic notation of this relationship is shown below:



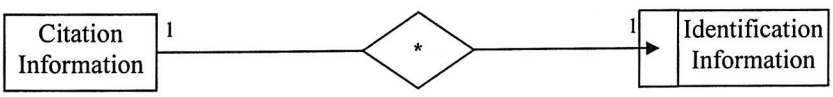
This is why the foreign key, bg_ii_id, of the *Browse_Graphic* table references the primary key, ii_id, of the I-I table. Figure 3.1 shows all the primary keys and foreign key for each metadatabase table.

Since the *Identification_Information* table is a requirement for the metadatabase, it is designed with a Primary Key labelled ii_id and four Foreign Keys connecting the tables *Citation_Information*, *Metadata_Reference_Information*, *Distibution_Information* and *Spatial_Data_Information.*

Citation related information specifies the recommended references used for the data set and its sources. The following notation represents the production rule for this entity. A more detailed description of individual elements is available in the next sub-section.

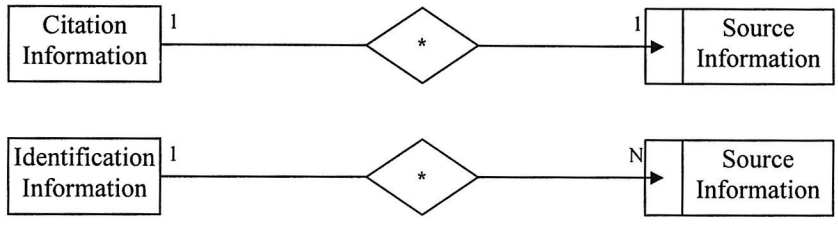
Citation_Information =
1{Originator}n + Publication_Date + (Publication_Time) + Title + 0{Edition}1 + 0{Geospatial_Data_Presentation_Form}1 + 0{Series_Information}1 + 0{Publication_Information}1 + 0{Other_Citation_Details}1 + (1{Online_Linkage}n) + 0{Larger_Work_Citation}1

There is a one-to-one relationship between the I-I and *Citation_Information* tables.



The notation above specifies that membership is obligatory for the *Identification_Information* table. Therefore, the foreign key, ii_ci_id, in the I-I table references the primary key, ci_id, in the *Citation_Information* table. A data set that is referenced by a citation must have an associated row within the I-I table.

However, the opposite may not be so, albeit it is an unlikely situation. *Citation_Information* is modelled as a separate entity because data sets often have source information, which must also be referenced (i.e. cited), as part of a metadata record. The CSDGM standard allows spatial data-providers with the option to specify information on data lineage by specifying events, parameters and source data used in constructing the data set. This information is contained within the *Source_Information* table which is related with the *Citation_Information* and I-I tables as follows:



Data set lineage is represented in the metadatabase using the *Source_Information* table. The production rule for source information was extracted from a portion of the Data Quality Information section of the CSDGM standard. Lineage information as implemented in the metadatabase is as follows:

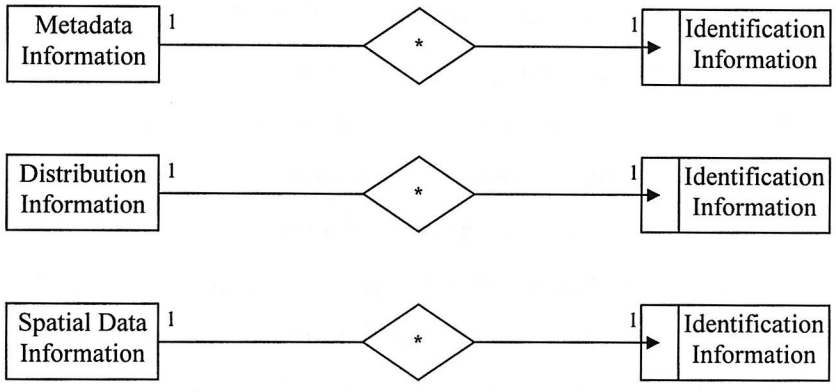
Source_Information =
Source_Citation + *0{Source_Scale_Denominator}1* + *Type_of_Source_Media* + *Source_Time_Period_of_Content* + *Source_Citation_Abbreviation* + *Source_Contribution*

Source related information of a data set is cited by a reference, hence the one-to-one relationship between the *Source_Information* and *Citation_Information* tables. So the foreign key, si_ci_id, of the *Source_Information* table references the primary key, ci_id, of the *Citation_Information* table. Membership is obligatory for the *Source_Information* table because citation records can either be associated with the I-I table or the *Source_Information* table but not both. Sources, on the other hand, must be properly cited with an originator, title, publisher etc.

Lineage information for a data set can come from more than one source (Lanter, 1991; Lanter and Surbey, 1994). Therefore the metadatabase must specify a one-to-

many relationship between the I-I table and *Source_Information.* Again, membership for the source information is obligatory because, in addition to having more than one source record for a particular I-I record, each source record must at all times be associated with an I-I record. Therefore the foreign key, si_ii_id, in the *Source_Information* table references the primary key, ii_id, in the I-I table.

Three other one-to-one relationships occur between the I-I table and the *Metadata_Infomation*, *Distribution_Information* and *Spatial_Data_Information* tables. These relationships are drawn below:



All of these relationships are membership obligatory, which means that any record entries in either of the tables must reference a record in the I-I table.

In order to enhance software development within the required time frame, it was necessary to considerably simplify the metadatabase by incorporating the Spatial Data Organisation Information, Spatial Reference Information and Entity and Attribute Information sections of the CSDGM standard into a single metadatabase table called *Spatial_Data_Information.* The CSDGM section specifying Entity and Attribute Information describes the entity types, their attributes and the domains under which attribute values are assigned.

Spatial Data Organisation Information consists of the mechanism used to represent spatial information in the data set. This includes the system of objects used to represent space as well as the name of types of geographic features, addressing schemes and other means through which locations are referenced in the data set. Collectively, such information consists of two broad categories that are designated as compound elements within the CSDGM standard. The first category includes Point

and Vector Object Information describing the types and numbers of vector or non-grid point spatial objects in the data set. The second category includes Raster Object Information describing the types and numbers of raster spatial objects in the data set.

The section within the CSDGM standard specifying Spatial Reference Information is used to describe the reference frame and encoding method of co-ordinates in the data set. It first specifies a Horizontal Co-ordinate System Definition comprising the reference frame or system from which linear or angular quantities are measured and assigned to the position that a point occupies. This can be geographic in nature where quantities of latitude and longitude define the position of a point on the Earth's surface with respect to a reference spheroid. Such information is represented by metadata elements such as Latitude Resolution, Longitude Resolution and Geographic Co-ordinate Units. Then there are compound elements that are planar in nature where the quantities of distances, or distances and angles define the position of a point on a reference plane on which the surface of the Earth has been projected. There are three broad planar categories, which include the compound elements of Map Projection Name, Grid Co-ordinate System and Local Planar (see Federal Geographic Data Committee (1994) for a detailed definition of all these metadata elements). Spatial Reference Information also specifies metadata elements that define the Vertical Co-ordinate System that includes the reference frame or system from which vertical distances are measured such as altitudes and depths.

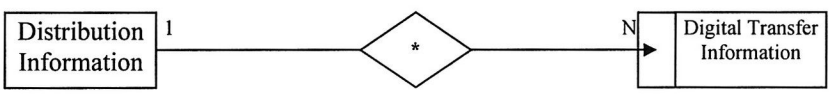
The *Spatial_Data_Information* table possesses a one-to-one relationship with the *Identification_Information* table. This means the foreign key, ii_sdi_id, in the I-I table references the primary key, sdi_id, in the *Spatial_Data_Information* table. All the elements that were chosen from these three CSDGM sections comprising the Spatial_Data_Information table are on the basis of a one-to-one relationship with the I-I table.

Perhaps one of the most important metadata entities with regard to a project that explores different methods of accessing metadata over the Web, is the Distribution Information section of the CSDGM standard. Distribution related information is stored in a table called *Distribution_Information*. It is a relatively small table where information about the distributor of the data set is kept. Various terms and

conditions for data access are also specified. The production rule for this table is described below:

Distribution_Information =
1{Distributor + 0{Resource_Description}1 + Distribution_Liability + 0{Standard_Order_Process}n + 0{Custom_Order _Process}1 + (Technical_Prerequisites) + (Available_Time_Period) }n

Notice *Standard_Order_Process* can be repeated 'n' number of times implying that there can be more than one method of accessing the geospatial data or literally none at all. There is a one-to-one relationship between the I-I table and *Distribution_Inforamtion* tables. The rest of the data-access information is stored in a separate table called *Digital_Transfer_Information* that is tied to the former by a one-to-many relationship. The relationship is symbolised through the following diagrammatic notation:



The one-to-one relationship above ensures that for every metadata record there is only a single recorded entry into *Distribution_Information* table. In reality, a data set can have more than one distributor. However, in order to reduce complexity, it was decided to limit this to just a single distributor per metadata record. That said, it would be logical to assume that a single data set (as represented by a metadata record) can have more than one method of access. Hence the existence of the *Digital_Transfer_Information* table that has a one-to-many relationship with the *Distribution_Information* table. Its production rule is as follows:

Digital_Transfer_Information =
Format_Name + ([Format_Version_Number | Format _Version_Date] + (Format_Specification)) + (Format_Information_ Content) + 0{File _Decompression_Technique}1 + (Transfer_Size)

The *Distribution_Information* table allows the distributor to specify one or more methods of digital data access. The one-to-one relationship between the I-I and *Distribution_Information* table requires membership obligatory for the I-I table. This means that each metadata record within the metadatabase must have a distributor describing the transfer of digital data. The relationship is implemented with the foreign key, ii_di_id, in the I-I table referencing the primary key, di_id, in the *Distribution_Information* table. The one-to-many relationship between *Distribution_Information* and *Digital_Transfer_Information* tables is membership obligatory for the latter table. The relationship specifies the foreign key, dti_di_id, of *Digital_Transfer_Information* references the primary key, di_id, of *Distribution_Information.*

The CSDGM standard reserves a separate section for information concerning the current nature of the metadata information as well as its authors. The metadatabase models the CSDGM compliant metadata information as a table labelled as *Metadata_Reference_Information.* The production rule that defines this entity is as follows:

Metadata_Reference_Information =
Metadata_Date + (Metadata_Review_Date + (Metadata_Future_Review_Date)) + Metadata_Contact + Metadata_Standard_ Name + Metadata_Standard_Version + 0{Metadata_Time_Convention}1 + (Metadata_ Access_Constraints) + (Metadata_Use_Constraints) + (Metadata_ Security_ Information)

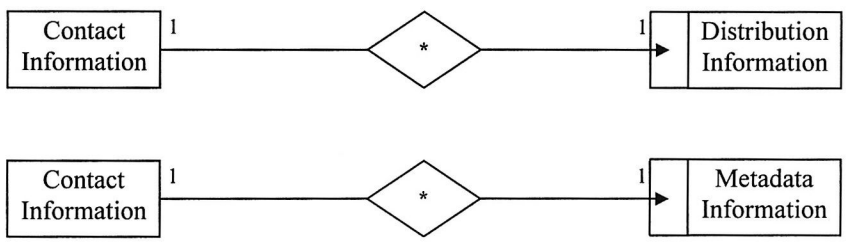
To reiterate, there is a one-to-one relationship between the I-I and *Metadata_Reference_Information* tables as shown in Figure 3.1. The relationship is membership obligatory for the I-I entity because for every metadata record there should be at most one record of information regarding the metadata file itself. Therefore the foreign key, ii_mri_id, in the I-I table references the primary, mri_id, in the *Metadata_Reference_Information* table.

What is notable is that both the *Distribution_Information* and the *Metadata_Reference_Information* tables require the identity and referral information of persons or organisation associated with the data set or its metadata to be explicitly

specified. This information is within a separate CSDGM section known as contact information which adheres to the following production rule.

Contact_Information =
[Contact_Person_Primary | Contact_Organization_Primary] + (Contact_Position) + 1{Contact_Address}n + 1{Contact_Voice_Telephone}n + (1{ Contact_TDD/TTY_Telephone}n) + (1{Contact_Facsimile_Telephone}n) + (1{ Contact_Electronic_Mail_Address}n) + (Hours_of_Service) + (Contact_Instructions)

Contact related information is stored in the metadatabase using a table called *Contact_Information* that has a one-to-one relationship with *Metadata_Reference_Information* and *Distribution_Information* tables, respectively. The diagrammatic notations is as follows:



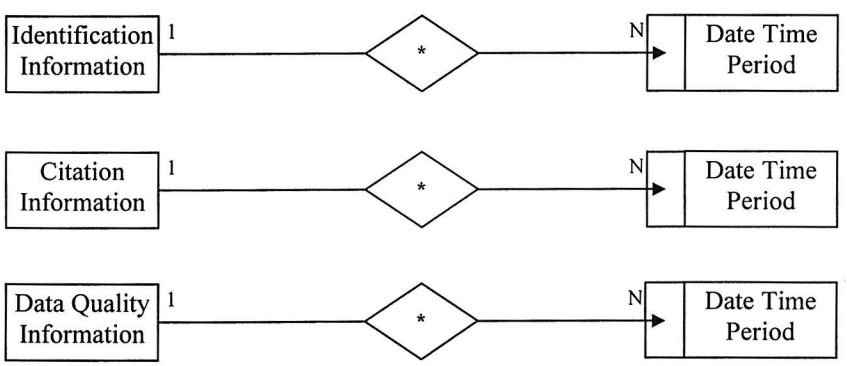
In both cases, the relationship is membership obligatory for *Distribution_Information* and *Metadata_Reference_Information* because contact information is an inclusive property for either of these tables and therefore, does not exist within the metadatabase as an entirely separate entity. That is why the foreign keys, mri_cti_id and di_cti_id, of these two tables reference the primary key, cti_id, of the *Contact_Information* table.

Keywords and Dates/Times are handled in a different manner than the metadatabase tables previously described. This is primarily due to the nature of the relationship these entities exhibit with other tables in the metadatabase. The CSDGM standard reserves a separate section to handle date and time information about any event that occurs with respect to a data set. This section is described by the following production rule:

Time_Period_Information =
[Single_Date/Time | Multiple_Dates/Times | Range_of_Dates/Times]

The CSDGM standard provides facilities to store individual date/times, multiple dates/times and ranges of dates/times. The Dates/Times section was implemented in the metadatabase as the table labelled *Date_Time_Period.* There are three metadatabase tables that store 'event' related information within the *Date_Time_Period* table; these include *Source_Information*, *Identification_Information* and *Distribution_Information.* These tables have a one-to-many relationship with the *Date_Time_Period* table with membership of either entity as non-obligatory. So it is not mandatory that any of the three tables have event-related information for each of their records. Such information can effectively be left out if the values are not available without compromising the referential integrity of the database. In turn, event-related information can be stored in the *Data_Time_Period* table without requiring it be related with the *Source_Inforamation*, I-I or *Distribution_Information* tables; though such a possibility would not make much sense.

The one-to-many relationships described with regard to the *Date_Time_Period* table operates through a series of 'Lookup' tables, one for each of the three metadatabase tables that utilise event-related information. The lookup tables store the primary keys of the two tables it is required to relate. Diagrammatic notation for these three one-to-many relationships is shown below.



So for example, a one-to-many relationship between the I-I and *Date_Time_Period* tables has a lookup table labelled ii_dtp_map that stores the primary keys of the I-I and *Date_Time_Period* tables, ii_id and dtp_id respectively. These lookup tables

provide a means to retrieve all the event-related records from the *Date_Time_Period* table associated with a particular metadata record. The same is true for the one-to-many relationships between the *Source_Information* and *Distribution_Information* tables and the *Date_Time_Period* table.

The CSDGM standard provides a formal structure to specify words and phrases used to describe various aspects of the data set. There are four categories. Theme keywords formally and informally describe various subjects and categories associated with the data set. Place keywords specify the geographic location of the data set and stratum keywords concentrate on layered or vertical locations within either a geologic or soils category. Temporal keywords indicate the time period of the data set. Each of these keyword categories have two types of keywords, one that is formally registered within a thesaurus and the other that is not. Registered thesaurus keywords were kept out of the metadatabase implementation because in the overall design of the GMS software, particularly the search-engine, such precise terminology was not needed. All four keyword categories are actually part of the Identification Information section of the CSDGM standard with the following production rules.

Keywords =
Theme + (Place) + (Stratum) + (Temporal)

Theme =
1{Theme_Keyword_Thesaurus + 1{Theme_Keyword}n }n

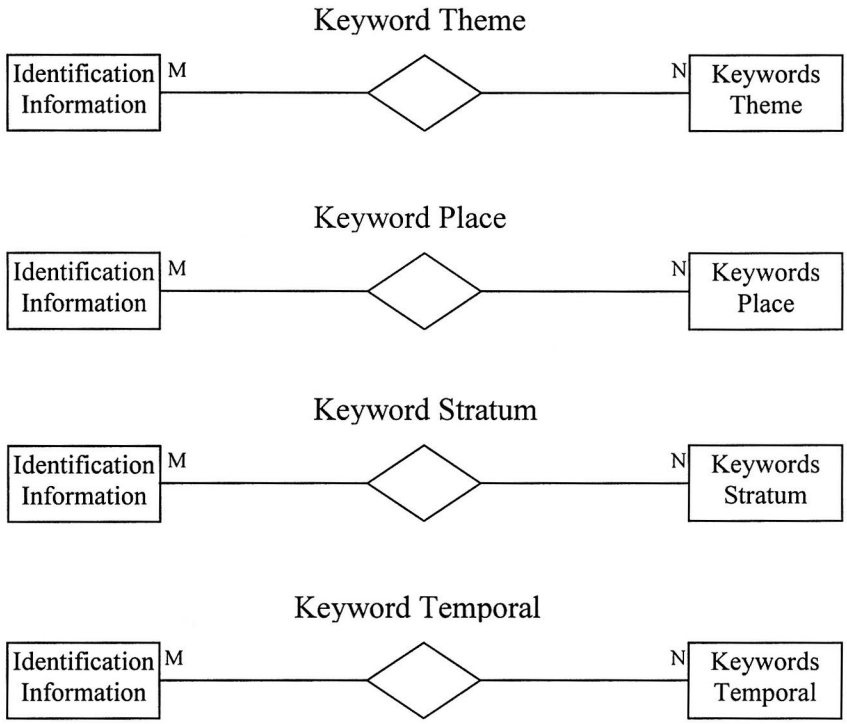
Place =
1{Place_Keyword_Thesaurus + 1{Place_Keyword}n }n

Stratum =
1{Stratum_Keyword_Thesaurus + 1{Stratum_Keyword}n }n

Temporal =
1{Temporal_Keyword_Thesaurus + 1{Temporal_Keyword}n }n

In all categorical cases, more than one keyword can be defined for a single data set. There is also the possibility that a single keyword (regardless of which category it is from) can be associated with more than one data set. The only means by which

entity-relationship modelling can show such an association between each of the keyword entities and the I-I entity is through a many-to-many relationship.



Theme, place, stratum and temporal keywords are placed in the metadatabase tables *Keyword_Theme*, *Keyword_Place*, *Keyword_Stratum* and *Keyword_Temporal* respectively. Keywords are mapped onto specific data sets and also through four different look-up tables, one for each keyword table. These look-up tables include *Keyword_Theme_Map*, *Keyword_Stratum_Map*, *Keyword_Place_Map*, and *Keyword_Temporal_Map*. So, for example, the key_theme_id and ii_id columns references both the primary keys of the *Keyword_Theme* table (key_theme_id) and the I-I table (ii_id), respectively, in the *Keyword_Theme_Map* table. The three other keywords are modelled in the same way.

The design considerations for the metadatabase are based on those sections deemed to be essential for the proper implementation of subsequent software applications. The design also aims to maintain a level expressiveness of the metadata records that captures the essence of the CSDGM standard without making the database overtly complicated.

The remainder of this sub-section focuses on some of the columns comprising the tables that were identified above as required components of the metadatabase. A formal definition of all the CSDGM metadata elements are not given in this account; rather the reader is referred to Federal Geographic Data Committee (1994) for more information in this regard. Metadata elements that are not mentioned in this sub-section can be referenced more thoroughly within the official documentation of the CSDGM standard.

The *Identification_Information* table contains the latitude and longitude of four bounding co-ordinates that delineate the western, eastern, northern and southern most co-ordinates of the geospatial data set. Longitudinal co-ordinates range from -180.0 degrees West to +180.0 degrees East while latitudinal co-ordinates range form -90.0 South to +90.0 North. Bounding co-ordinates such as these are of particular importance to geographically related search queries, which are used in on-line maps and atlases. This table also contains the columns *Full_Name*, *Password* and *Access#* that identifies one or more geospatial data sets with a particular username and password as well as an access number to determine system privileges for a particular user and/or data set.

The *Browse_Graphic* table contains four columns that store metadata on graphic files for the illustration of geospatial data sets. Information on name, description and type of graphic files are used in conjunction with the on-line linkage column (*BG_File_Online_Linkage*) to provide the ability to view these graphic files from any networked Web server. The *BG_File_Online_Linkage* column is a standard URL address.

As mentioned in the previous sub-section, the metadatabase handles keywords through many-to-many relationships between four separate keyword tables and the *Identification_Information* table. The four tables (*Keyword_Theme*, *Keyword_Temporal*, *Keyword_Place* and *Keyword_Stratum*) each consists of two columns, the primary key and the keyword elements. Only a single type of keyword is used for each table describing common-use words or phrases as well as those keywords formally registered in some authoritative source.

The metadatabase uses a single table for all date-related information particularly in regard to the one-to-many relationship between the *Date_Time_Period* table and the

source, identification and distribution information tables. With the one-to-many relationship, either single or multiple dates are stored in the *Calendar_Date* and *Time_of_Day* columns. A range of dates and times are stored in the *Date_Begin*, *Date_End* and the *Time_Begin*, *Time_End* columns. These metadata elements are important aspects of the GMS system as a whole because temporal information is a vital component in searching on-line catalogues of metadata. Often users require geospatial spatial data of a certain time period consisting of single date/time or a range of dates. It's a fact that the spatial phenomena that geospatial data represents or models change over time. The time spans involved can be extensive such as the case with geologic maps or relatively small as with maps that depict the changes that occur with the road network and urban occupation of a town, city or state. Maps showing historical and political data also change over time. Such diverse temporal information must be accommodated adequately within a metadata database. This is achieved because the design of the metadata database for date-related information adheres to the CSDGM standard.

The CSDGM standard specifies that the compound element, Digital Transfer Information, is contained within the Distribution Information section of the standard. Because of the one-to-many relationship between certain elements of Distribution Information and Digital Transfer Information, both are modelled using separate tables. For the *Digital_Transfer_Information* table, only those elements that are relevant to the network access of geospatial data were implemented. *Format_Name,* for instance, enables the GMS search-engine (Chapter 4) to determine the vendor specific format of the geospatial data thereby allowing it to assign the correct data-conversion utilities. If the data set is compressed, then the *File_Decompression_Techique* column specifies the compression algorithm that should be used to decompress the file on the fly. *Transfer_Size*, *Network_Resource_Name* and *Network_Address* columns are all vital pieces of information for GMS to access the geospatial data set from any Web server.

The database is implemented using a single SQL macro located on the CD-ROM, appendix A. Since the database underwent considerable modification and testing during software development, it was deemed necessary to again re-implement the

database every time such modifications were made. These modifications were only possible by dropping all the previous references of tables within the database. Hence, the statement below was executed for each database table.

DROP TABLE <schema.> <table_name> <cascade_constrants>

Followed by,

CREATE TABLE <schema.> <table_name> <table_constraint>
where <table_constraint> = <column_datatypes> <column_constraints>

In order to create each of the database tables.

The SQL commands SELECT, UPDATE, INSERT and DELETE operate on each of the database tables through the HTTP server. It was necessary to grant each of the tables appropriate system privileges. This was accomplished using the following command.

GRANT <system_privileges | roles> ON <table_name> TO <user | role | public>

In this case, the HTTP server (ops$http_server) is allowed to execute the four SQL commands on a specified database table.

In order for GMS to insert new database records into the metadatabase, a SQL function was needed to automatically generate primary key values for each of the database tables. This is accomplished by a *sequence*, which is a database object that allows multiple users to generate unique integers for use in primary key values (ORACLE7™ Server, 1992a). Sequences are created by the following command.

CREATE SEQUENCE <schema.> <table_name>

Sequences must be dropped in order for them to be re-sequenced (i.e. begin from 1) and all such database objects much be allowed access to the HTTP server.

3.1.5 Design and implementation constraints

The Content Standards for Digital Geospatial Metadata was the primary guide used to model the metadatabase and it was this specifications that provided the main constraint in the design of the database tables, columns and relationships. The National Spatial Data Infrastructure NSDI encompasses policies, standards, and procedures for organisations to co-operatively produce and share geographic data among 15 federal agencies, state/local/tribal government, academic community and the private sector. Because CSDGM is an United States standard, it primarily co-ordinates and standardise the description of geospatial data within US organisations.

Refined drafts of the CSDGM standard were released for review and testing as early as January of 1994 making it the only complete and comprehensive metadata standard for geospatial data available at the time. Because of Executive Order 12906 all federal agencies were required to document new geospatial data using the CSDGM standard and provide access to this document via the FGDC Clearinghouse network (Clinton, 1994). As a result of this concerted effort by the US government, a considerable amount of standardised metadata and referenced geospatial data was already accessible for testing purposes during the development of the GMS metadatabase. The use of standardised metadata and geospatial data is very effective in demonstrating system functionality within prototype systems such as GMS. Because the metadatabase was designed using the ORACLE7™ RDBMS, it is subject to all the limitations associated with that software package. The metadatabase was written in standard SQL and should be portable to other DBMSs that support this standard API for databases.

3.2 Metadata parsing and database entry

This section describes a system that uploads a CSDGM compliant metadata file, parses it for metadata by comparing the information to pre-specified metadata elements and loads the information into their proper metadatabase tables and columns. The parsing program is called MPARSER; it was developed as a stand alone utility designed to read CSDGM compliant metadata files and reorganise the information into a simpler format for database entry. The database entry program, MLOADER, reads the parsed metadata file and inserts the information into the proper tables and columns of a RDBMS or metadatabase as described in the previous section. MLOADER consists of a number of smaller programs, which are designed to operate on a specific metadatabase table while maintaining the referential integrity of the database (Figure 3.2).

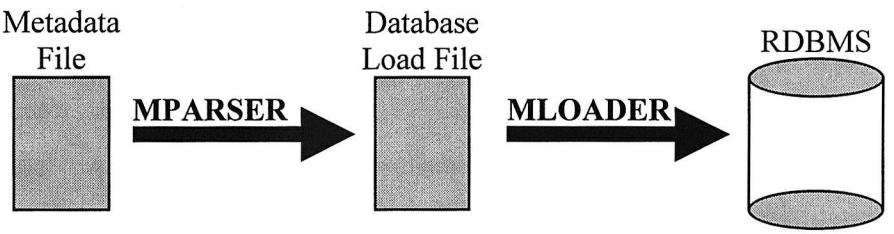


Figure 3.2 MPARSER and MLOADER within GMS.

Some geospatial data dissemination sites use certain types of software designed to automatically parse a metadata file for the purpose of on-line catalogue and storage. In this context, parsing of information such as metadata is a form of analysis through its disassembly in order to recognise its constituent components (Schweitzer, 1996a). The process of parsing a metadata file is to separate the information associated with each element from that of other elements while maintaining some sort of association between the section names, element names and element values. Usually the parsed information is stored in another file that is more easily readable by other applications in the system.

The DIF loader is an integral part of the Global Change Master Directory because it provides a relatively fast and effective means to insert automatically metadata into an on-line metadata database. The loader reads each line of a DIF text file, compares

each DIF field to a syntax dictionary and translates them into ANSI standard SQL statements for database entry. DIF field values are parsed from a DIF text file and read into Field-ID. SQL statements validate these field values, which checks if they are suitable for database entry. Validation is followed by the creation of SQL insert statements that direct the SQL interpreter to insert the DIF values in the proper tables and columns of the RDBMS (Vogal, 1996). The DIF loader, used within the GCMD system, is perhaps one of the best examples of on-line parsing and database entry software.

The GMS metadata parsing and database entry utility is somewhat similar to the one used for the Global Change Master Directory with the important exception that the former uses the CSDGM standard to store metadata while the latter uses the DIF standard. Another important difference is that DIF loader validates metadata using SQL statements before entering it into the database. The GMS parsing and database entry utilities, MPARSER and MLOADER, perform metadata validation more implicitly by checking the form and structure of the metadata within the Pro*C programs. The values of the metadata elements are checked so that they conform to the data types (DATE, NUMBER and VARCHAR2) of the database column for which they correspond. The result is a substantial performance advantage because the GMS metadata parsing/entry utility uses fewer SQL statements for validation and database entry than the GCMD loader.

Site-administrators of on-line metadata and geospatial-data dissemination sites use parsing programs to maintain the currency, accuracy and quality of metadata entered into the database. Data-providers can also quickly build their own comprehensive centralised metadata catalogues. Standalone parsing software can be used to validate quickly metadata, which can then be uploaded as a flat file to a server or entered into an on-line database. The Metadata Parser or mp (Schweitzer, 1996b) is one such program which validates hierarchically-indented textual metadata against the CSDGM standard. This utility as well as a series of AML creation tools (Negri, 1996; Katz, 1996; Daumiller, 1996; and Nelson, 1997) are designed to be portable and relatively simple for site-administrators to install and use in order to check the accuracy and quality of the metadata added to their sites. However, these parsing utilities can not be easily integrated with existing software on the Web site without

writing additional supporting code. This fact precludes their use within on-line metadata entry utilities. If the parsing programs can not be easily implemented on-line, then there is no possibility that data-providers located throughout the Internet can use their capabilities to read metadata and submit it to an on-line catalogue.

Both the GMS metadata parsing and database entry utilities are implemented on-line as integral system components so as to ensure access to users anywhere on the Internet. The MPARSER and MLOADER utilities provide the means for site-administrators to enter quickly and validate metadata files thereby maintaining the currency of the metadatabase as well as its accuracy and quality. Also, both on-line utilities are easily accessible to data-providers who can submit their metadata as text files. These files can be subsequently parsed and entered into the metadatabase to build the metadata catalogue. Neither program is easy to use but they potentially reduce the time it takes the user to enter a CSDGM metadata file into a RDBMS. Given these metadata files are usually several pages long, entering one into the database by hand would literally take hours and asking someone to type it into a WWW forms or Java interface would be just as intolerable.

The remainder of the section outlines the design of the metadata parsing/database entry utility through a series of flow diagrams. The diagrams are followed by a detailed description of the software implementation for the MPARSER and MLOADER programs.

3.2.1 Software design

A single CGI script (Deep and Holfelder, 1996) is used to control both the parsing (MPARSER) and database entry (MLOADER) programs. Figure 3.3 is a flow diagram of the script whose primary responsibility is to upload metadata files, parse the files of information, extract appropriate arguments and supply them to various on-line programs. The utility either associates appropriate on-line software with the information contained in the parsed metadata file (see MSOFT program described in appendix D) or inserts the metadata into the correct tables and column of a relational database.

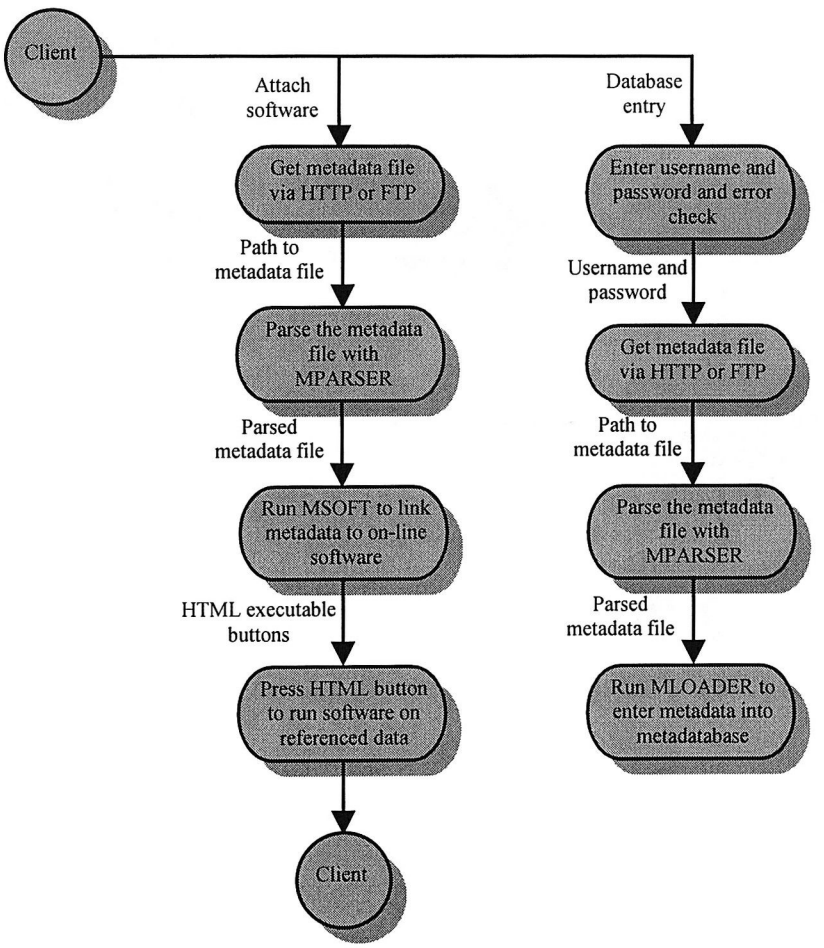


Figure 3.3 Flow diagram for the database entry and software access utility. The diagram to the left executes the MSOFT utility (appendix D) while the right executes the MPARSER and MLOADER utilities one after another.

In both cases, the metadata files are parsed for information. The primary steps the MPARSER program uses to parse section names, field names and metadata values from a CSDGM metadata file are shown in Figure 3.4. Appendix E provides a sample CSDGM metadata and the associated output after it is run through the MPARSER program.

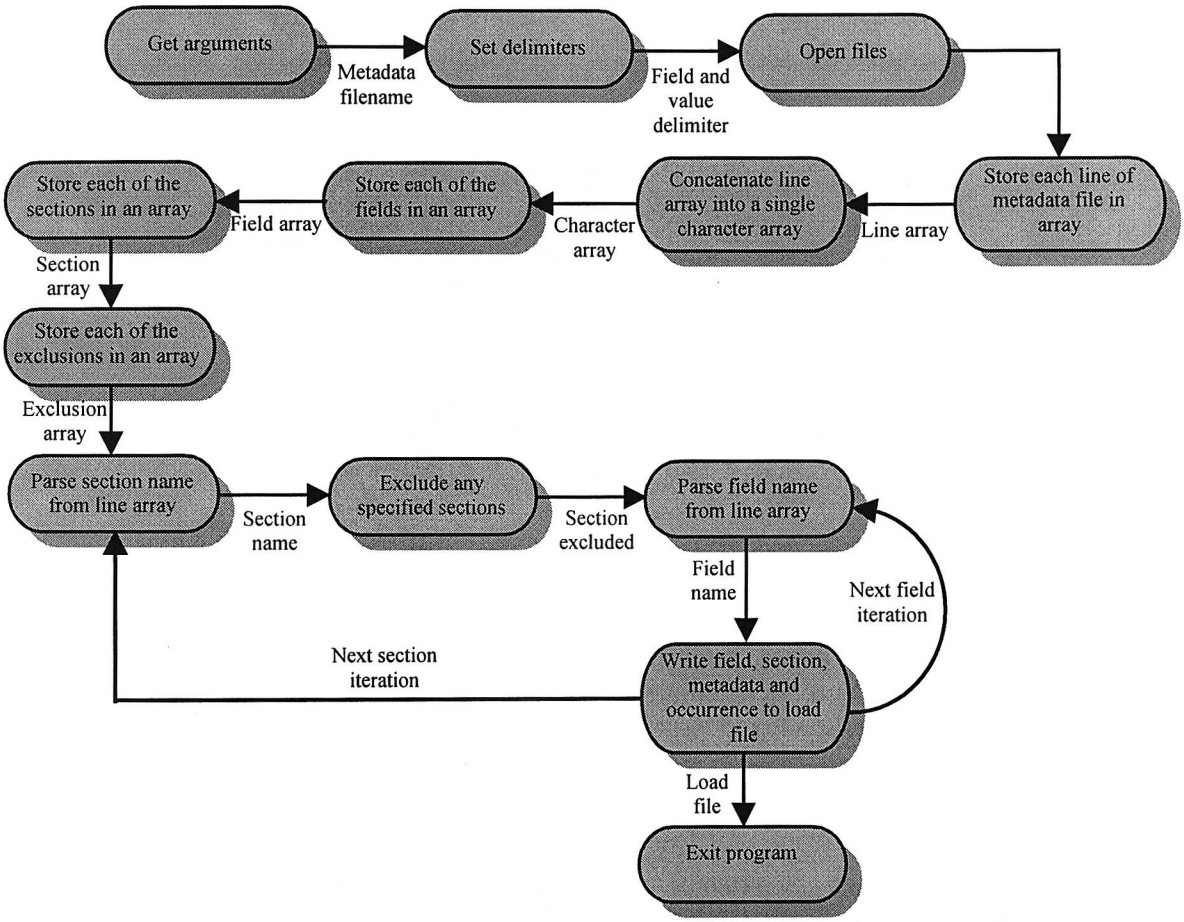


Figure 3.4 Flow diagram for the MPARSER utility.

If the user selects the database entry option, the metadata file is parsed and the MLOADER utility is executed. Figure 3.5 is a flow diagram of the MLOADER utility showing the master.pc program, which controls and executes three sets of other programs with each set containing fourteen programs or one for each table in the metadatabase. The three sets of programs parse the load file for their respective metadata elements and values (Figure, 3.6), insert the metadata values in their proper database tables and columns (Figure, 3.7) and finally free any memory reserved during the process.

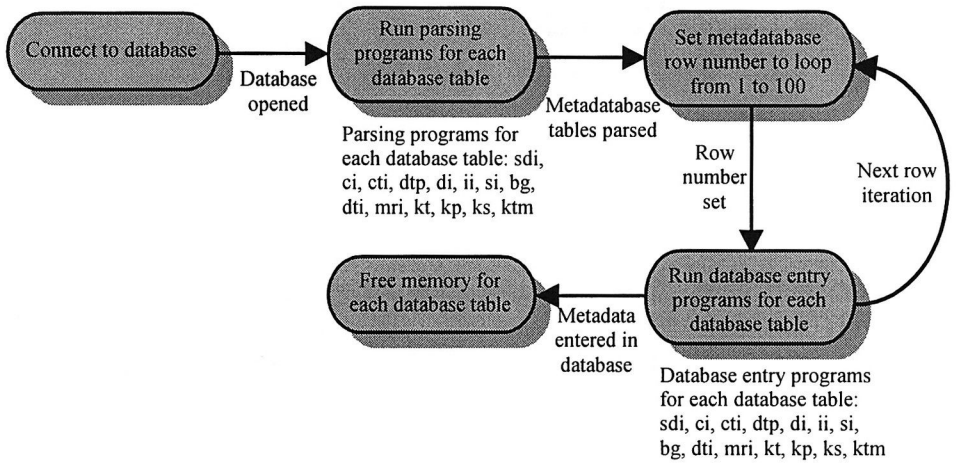


Figure 3.5 The master program for the MLOADER database entry utility. Each metadatabase table is labelled with an acronym (e.g. ii for *Identification_ Information*). The operation of each parsing program is generalised by the diagram of Figure 3.6. The operation of each database entry program is generalised by the diagrams of Figure 3.7.

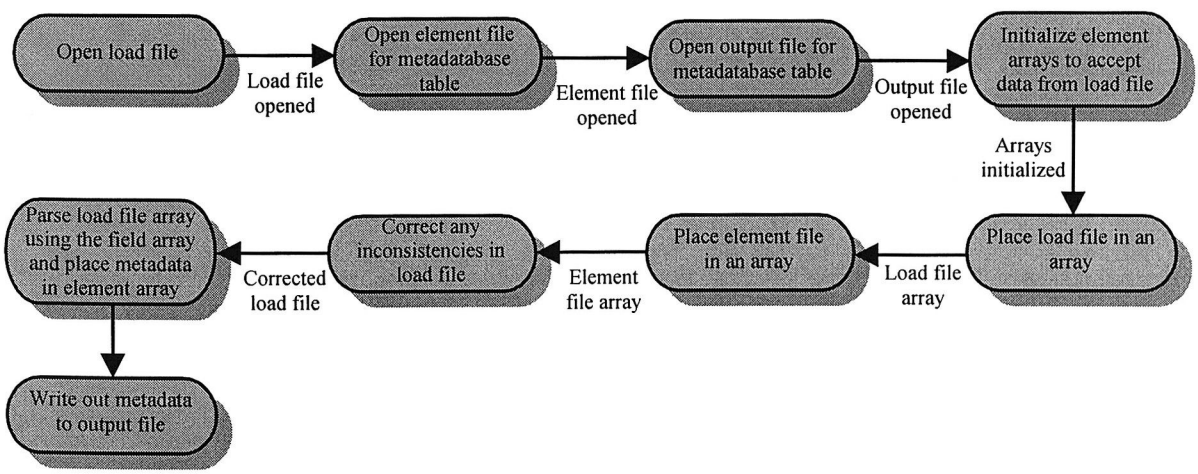


Figure 3.6 The parsing program constructed for each metadatabase table.

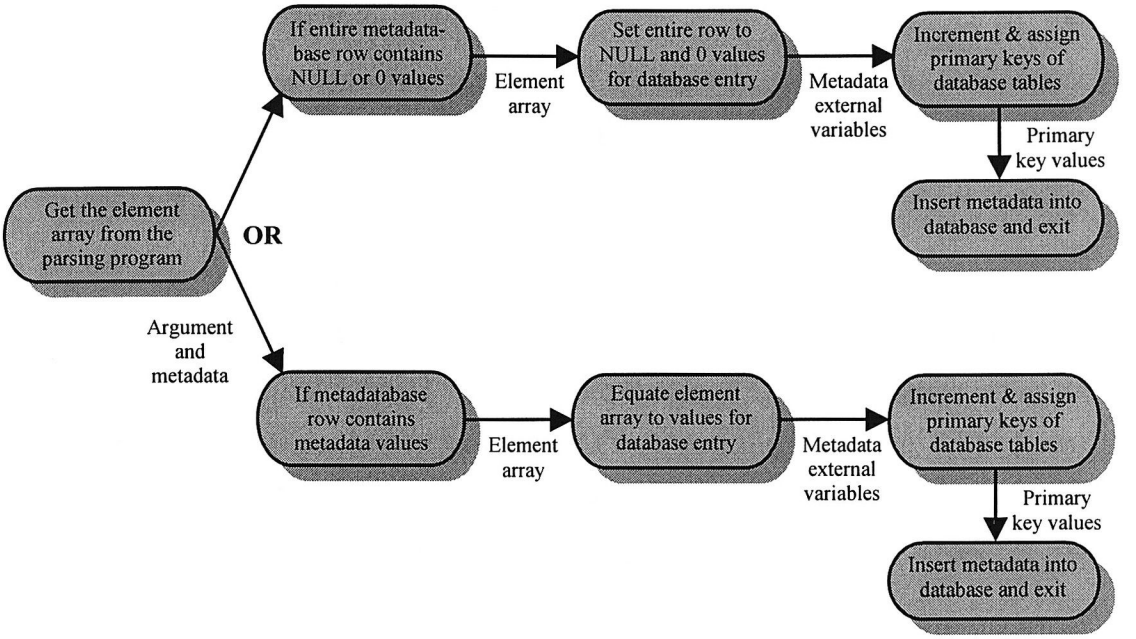


Figure 3.7 The database entry program constructed for each metadatabase table. The above branch is run if the row contains no metadata values. This is done to ensure primary and foreign keys are incremented, which maintains the referential integrity of the database.

3.2.2 Software implementation

The next three sub-sections describe in detail the implementation of the CGI script, MPARSER program and MLOADER program, which makes up the entire metadata parsing and database entry utility.

3.2.2.1 The user-interface and CGI script

The user-interface for the metadata parsing and database entry utility consists of a single HTTP or FTP upload field and two radio-buttons. The radio buttons allow the user to operate either the MSOFT or the MLOADER utilities (Figure, 3.8). Recall that the MSOFT utility 'attaches' metadata to the appropriate on-line software while the MLOADER enters metadata into the proper columns and tables of the metadatabase.

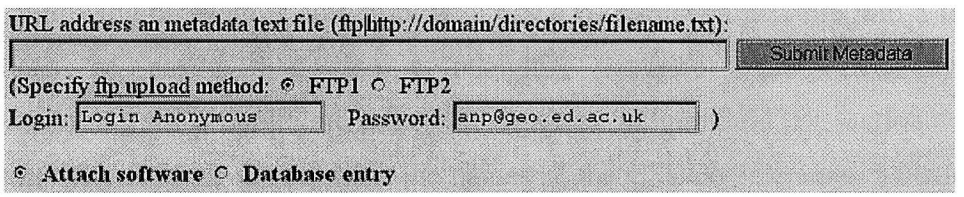


Figure 3.8 The user-interface for the metadata parsing and database entry utility. User can upload a text file containing metadata and process it using the MSOFT or MLOADER utilities via the radio buttons, respectively.

Upon pressing the 'submit metadata' button, the form field arguments are sent to the CGI script shown in Figure 3.3. The database entry utility accepts the arguments and prints two HTML text fields to the browser prompting the user to enter a username and password. As stated in the previous section of this chapter, metadatabase rows are identified with a row number, username and password contained within the *Identification_Information* table. When a username and password is submitted to the database entry utility the data is passed to the MLOADER program which generates a new database row containing the user's identification and security code. However, prior to database entry, the CGI script uploads the metadata file using the given URL address and its directory path is sent to the MPARSER program. The metadata file is subsequently parsed of its metadata sections and elements; the information is then entered into the proper database tables and columns by MLOADER using the newly generated primary key of the I-I table.

To attach on-line software to information parsed from a metadata file, the file is initially uploaded and its directory path is sent to the MPARSER program. The MSOFT program examines the parsed metadata, selects key metadata elements and sends their values to on-line software as command line arguments. On-line software (e.g. the conversion utilities or the Web robot) is executed given the information supplied by the MSOFT program. The metadata elements that MSOFT program looks for while parsing a metadata file are specified in a separate text file which the program reads when executed. The CGI script prints one or more HTML executable button(s) to the Web browser containing (within hidden fields) all the arguments needed to execute the on-line software utilities cited by the MSOFT program.

3.2.2.2 The MPARSER utility

The following program is able to parse a CSDGM metadata text file into a format that can be easily read into a relational database management system. Accompanying the program are three text files that include field.txt, section.txt and excl.txt. The field.txt file contains the names of the CSDGM elements MPARSER uses to compare with the metadata elements found in the metadata file. The section.txt file contains the CSDGM section names MPARSER uses to keep track of its location while parsing the metadata file. Since some metadata elements are used in more than one section, the MPARSER program must know which section it is parsing to ensure that the metadata value of any particular element is placed in the proper metadatabase table. The excl.txt file indicates those sections the user wishes to exclude entirely as the metadata file is parsed. In order to exclude a particular section that contains all its elements and accompanying metadata, the section name must be included in both the section.txt and the excl.txt files. Indicating the sections to be parsed and the sections to be excluded ensures that only the required metadata is inserted in the database. By editing all three files, the user can direct MPARSER to parse only those section names and element names the user deems necessary to enter into the metadatabase.

The three files used by the MPARSER program adhere to a simple format. The field.txt file uses the string, "--", as a line separator that (ignored by the parsing routine) provides the user with a means to clarify the contents of the file. Figure 3.9

shows a line separator between the Source_Contribution and the Abstract metadata elements. Refer to the CD-ROM (appendix A) for the entire field.txt file.

```
**Identification Information
Originator
Publication Date
Publication Time
Title
Edition
Geospatial Data Presentation Form
Series Name
Issue Identification
Publisher
Publication Place
Online Linkage
Other Citation Details
Source Scale Denominator
Type of Source Media
Source Currentness Reference
Source Citation Abbreviation
Source Contribution
--
Abstract
Purpose
.
.
.
```

Figure 3.9 A portion of the field.txt file showing the metadata elements for the *Identification_Information* table.

It should be stated that the order the element names occur in the field.txt file is irrelevant because the program compares each metadata line with all the element names (of a particular section) for a match. Also notice that all the section names within the field.txt file are prefixed by the '**' characters. This prefix is a requirement because MPARSER uses it to identify section names within the field.txt file. The program essentially tracks section names within the metadata file and the field.txt file thereby substantially reducing the amount of processing. For example, if the program is currently parsing the section Distribution_Information in the metadata file, then it only needs to use the element names (in the field.txt file) of that section. All other element names are ignored.

The section.txt file simply lists all the CSDGM section names contained in the metadata file. The list can be reduced or expanded depending on the number of sections in metadata file. The contents of section.txt file used in developing and testing the MPARSER program is shown below:

Identification Information
Lineage
Process Step
Spatial Data Organization Information
Distribution Information
Metadata Reference Information

It is important that users include all the section names in their metadata files. Otherwise, the metadata occurring under sections that are not to be parsed may be inserted into database tables and columns that are originally designed to contain metadata from other sections of the metadata file. For example, most metadata files contain information about the events, parameters and source data used to construct the data set, which is a sub-section of Data Quality Information called Lineage. Information about a single event, known as a Process Step, is contained in a sub-section of Lineage.

2 Data Quality Information
2.1 Attribute Accuracy
2.2 Logical Consistency Report
2.3 Completeness Report
2.4 Positional Accuracy
2.5 Lineage
2.5.1 Source Information
2.5.2 Process Step

In order to exclude the metadata elements associated with Process Step yet retain the other Lineage related information, the Process Step sub-section is excluded from parsing by entering its name in the excl.txt file.

The MPARSER program operates by scanning a metadata text file line by line and concatenating all of the lines into one large string (Figure, 3.4). The program scans a standard ASCII text file and places each line into a string or array of characters. The strings are then concatenated into a single large character array to facilitate the process of parsing section and element names from the metadata document. The element names and metadata sections, as well as those sections excluded from parsing, are each placed in a separate array of strings.

The results of the parsing routine are outputted to both standard I/O and a text file indicated by argv[1]_loadfile.txt file name where argv[1] is the first argument on command line. This 'load file' has a format defined as follows:

```
Title
Identification Information
1
TIGER/Line Files, 1995
**********
.
.
.
```

Title -- CSDGM element name, a database column
Identification Information -- CSDGM section name, a database table
1 -- occurrence value in the load file, i.e. its possible to have more than one row in a particular table
TIGER/Line Files, 1995 -- the value of the element, metadata
********** -- a delimiter used by another program to parse the load file prior to database entry

The successful operation of the parsing program depends on the two program delimiters that are used in the metadata file. The delimiters can be manually changed in the declarations section of the program and the code subsequently recompiled. These program delimiters are as follows:

```
int database=10000;
char field_delimiter = ':';
char value_delimiter = '|';
```

The field_delimiter marks the end of a metadata element or section name and the value_delimiter marks the end of a metadata data value. So a typical delimited metadata file displays the following format:

```
Identification_Information:
   Citation
      Citation_Information
         Originator:
            U.S. Department of Commerce
            Bureau of the Census
            Geography Division|
```

Notice Citation and *Citation_Information* need not be delimited because they are not included in the field.txt and section.txt files as element and section names, respectively. In practice, only the elements that are parsed are delimited; if more information is delimited than is necessary, the excess information would also be inserted into the database alongside the required metadata. The choice of delimiters

is also crucial. A ':' delimiter may or may not create problems in parsing the document because strings such as "http://www..." and "8:30pm" use this particular delimiter. However, usually MPARSER does not have a problem with the ':' character because the program looks at the entire field and/or section name including the delimiter, (i.e. Originator:). Also, all the string comparisons conducted by the program are case insensitive.

It is good practice while delimiting a CSDGM metadata file to use a '_' character between section and field names that span more than one word, such as Identification_Information:, South_Bounding_Coordinate: or State_or_Province:. The parameter int database=10000, controls the size of the element value. For example, the largest datatype within the ORACLE7™ RDBMS used to develop this software is the VARCHAR2 with a maximum length of 2000 characters. Since the VARCHAR2 does not accept anything more than 2000 characters, the limit (i.e. int database=2000) is the maximum possible length of a metadata values that can be inserted into the metadatabase. The 'database' parameter ensures that only 2000 characters are parsed for each data value in the metadata file.

Most CSDGM metadata files are in excess of 20000 characters, so it could take MPARSER anywhere from 3 to 5 minutes to process the document. As a general rule, the more complicated the database (i.e. the more one-to-many and many-to-many relationships) the less accurate is the MPARSER parsing program.

The above commentary makes it easier to explain exactly how MPARSER extracts metadata sections, elements and data values from a CSDGM metadata file. The character array containing the entire metadata file is repeatedly traversed a character each time until all the metadata sections and elements indicated in the field.txt and section.txt files are completed. The entire character array is parsed for a CSDGM section equivalent to one of the section names within the section.txt file. Upon finding a section and verifying that the section name is not contained in the excl.txt file, the parsing program starts at the top of the field.txt file, accesses the element names for the particular section and searches for each element name within the character array. When an element name is found, the contents of the element is extracted (to the '|' value delimiter) and saved to the load file. As shown in Figure 3.4, each element name is parsed from the character array until the **section_name

in the field.txt file is reached. Then the iteration in the section.txt is advanced to the next section name and the entire process begins again. Only the element names in the field.txt file for a particular section are used for the iteration of the section loop. Figure 3.10 shows the relationship between the section.txt and field.txt files while the parsing program is in operation.

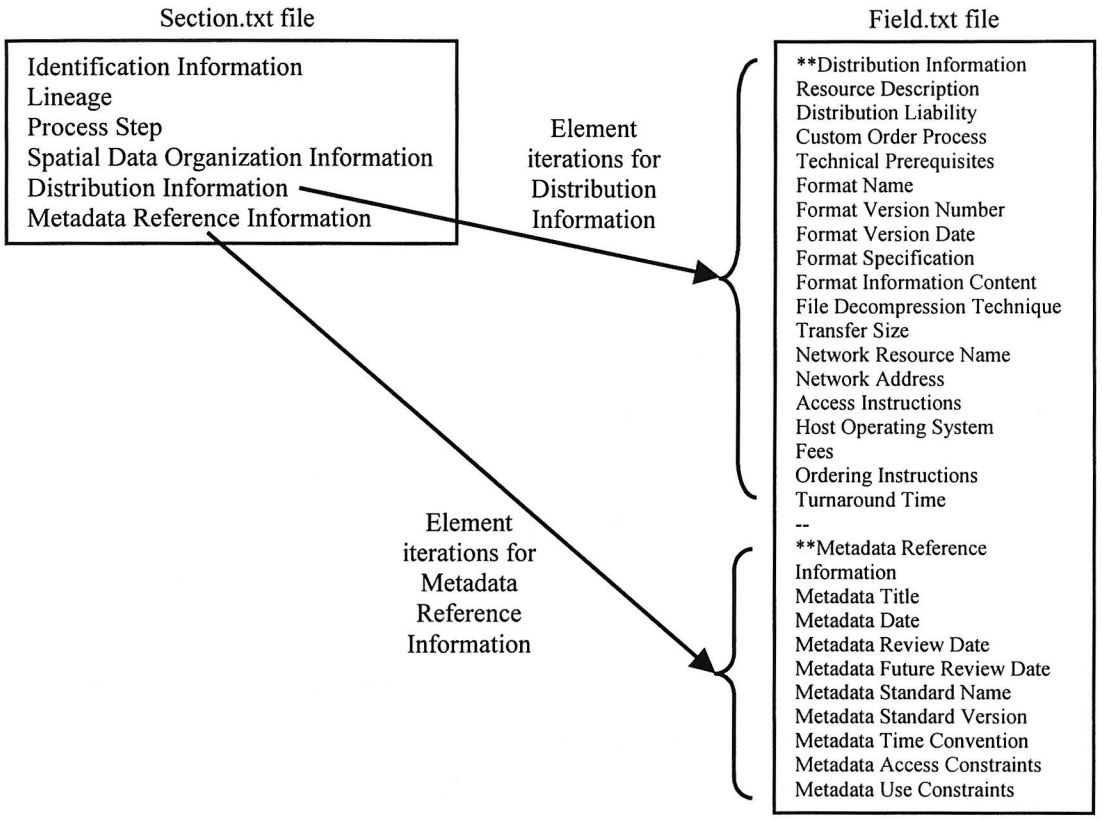


Figure 3.10 MPARSER program iterations for the section.txt and field.txt file.

3.2.2.3 The MLOADER utility

The primary purpose of the MLOADER utility is to read the metadata contained in the load file (produced by the MPARSER program) and inserts[2] the information into the proper tables and columns of the metadatabase while maintaining referential integrity of the database. MLOADER is composed of three sets of programs that parse the load file, enter the metadata into the database and free any allocated memory. There are fourteen programs in each set that correspond to each of the

[2] See Oracle® (1989) and Oracle® (1992) information on the Pro*C embedded-static SQL language and its associated precompilers.

fourteen-metadatabase tables. Each table in the database possesses a parsing program that searches the load file of the metadata element names that correspond to its own columns, extracts the metadata associated with the element name and inserts it into the database. Each table in the database has a database entry program that inserts the metadata into the proper columns of the table incrementing the table's primary key and updating the primary and foreign keys of other tables where necessary. The MLOADER program creates 'two-dimensional string arrays' for each column of each database table. Finally, each table possesses a program, which frees the memory of the 2D-arrays for each of its database columns.

The flow diagram of Figure 3.5 summarises the master.pc program that executes the parsing, database entry and memory freeing programs for each table of the metadatabase. The master.pc program declares the 2D-string arrays in a separate header file and controls the execution of all the other programs, particularly with regard to the set of programs responsible for data entry.

The master.pc program first opens the ORACLE database and runs the fourteen parsing programs; the order that these programs run is inconsequential. Each parsing program reads the same load file but searches for different element names depending on the database table affiliated with each program. For example, the parsing program designed for the Spatial_Data_Information table searches for metadata element names in the load file that match only the columns of this particular table. An element name such as 'UTM Zone Number' is used by the PSDI program (Figure, 3.5) to search the load file for an element that matches the same designation. If the element name is found in the load file, then the metadata associated with that field is stored in one of the 2D-arrays (**(utmnum+count)* to be exact) mentioned above. The DSDI program reads this array and enters the metadata value into the corresponding column of the Spatial_Data_Information table, UTM_Zone_Number. Figure 3.11 shows the entire process below:

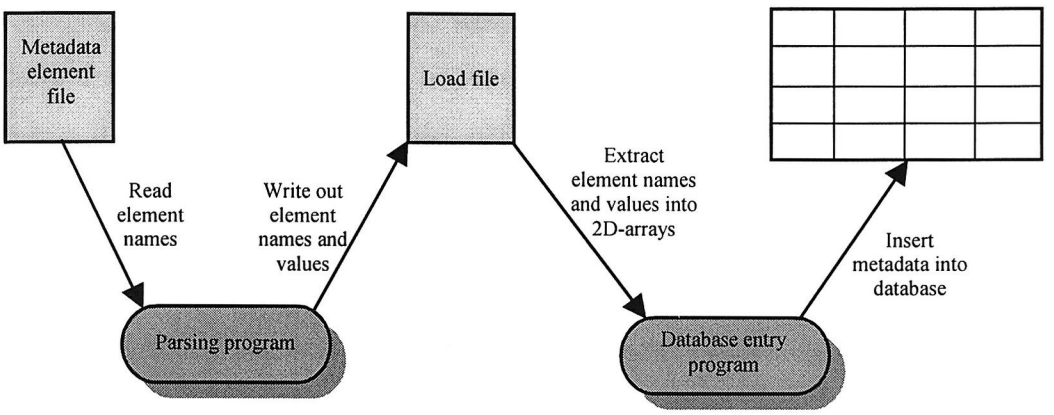


Figure 3.11 Extracting metadata from a metadata file and its subsequent database entry.

The parsing programs begin by opening three files that include a load-file containing the metadata, a text file containing the names of the elements and an output file to write the metadata extracted from the load file (Figure, 3.6). The output file is used more for error checking because in actuality, the extracted metadata is written directly to the database. The 2D-arrays designed to contain the metadata are allocated memory and initialised by default values that include 'NULL' strings for textual metadata and '0' values for numeric metadata. In C this is reduced as, where BUFFLEN is assigned a value of 2000:

```
*(abstract+count) = (char*)malloc(sizeof(char)*BUFFLEN);
*(pctcldcov+count) = (char*)malloc(sizeof(char)*BUFFLEN);
*(abstract+count) = "NULL";
*(pctcldcov+count) = "0";
```

Initialising these arrays ensures the database rows of each table are never empty and any particular row can be easily identified as with or without metadata. The 2D-arrays are declared in the MLOADER header file, metadata.h, as variable-length arrays of strings.

```
char *abstract[30];
char *pctcldcov[30];
```

The arrays are declared as pointers and allocated a maximum length of 2000 characters. There is a maximum of 30 possible arrays for each metadata element. Hence, the two dimensional length of 2000 and a width of 30 are used. Therefore, a single 2D-array, representing a particular element name, can store a maximum of 30

strings or metadata values each being a maximum of 2000 characters in length. With the exception of the place, stratum, temporal and theme keyword elements, it is unlikely the 30 string limit would be reached by the parsing programs in a single metadata file. The length is another matter, because the 2000 character limit is frequently exceeded by a number of metadata elements such as 'Abstract', 'Purpose', 'Entity and Attribute Overview', 'Resource Description' etc. If the limit is exceeded the data can not be inserted into the database. The limit is the maximum allowable using the ORACLE varchar2 external variable (see sub-section 3.2.3). Alternative arrangements must be made for metadata that exceeds this important length limitation. This includes using other ORACLE data types, which have their own limitations, or storing the metadata element in more than one database column.

The diagram in Figure 3.11 shows that the parsing programs access two files including a text file of metadata element names and a load file containing metadata organised by the MPARSER program. Both these files are read into character arrays (element file array and load file array, see Figure 3.6) for use in the MLOADER parsing program. Prior to parsing, any inconsistencies in the load file array are corrected so as not to enter metadata that could seriously harm the referential integrity of the metadatabase. Usually the inconsistencies occur with the 'occurance' value of the load file, which tracks the number of rows used to enter metadata values for any given table of a particular metadata record.

For example, a geospatial data set represented by a single metadata record (or row) can have numerous source data sets that contribute to its creation (i.e. one-to-many relationship). Each of these sources must have a citation as well. So, a single metadata record can have multiple entries in the *Source_Information* table and each source must have a corresponding entry in the *Citation_Information* table. Given a situation where a metadata record has, say, 4 sources, there are 'occurance' values for each source and its corresponding citation, number 1, 2, 3, 4 (see load-file in appendix E). Occasionally after parsing these numbers can be out of sync (i.e. 2 3 4 1 for the source and 4 1 3 2 for the citation). Since the MLOADER (see below) program enters information table by table for a particular record, it is possible that a source can be associated with the wrong citation or vis-versa. The MPARSER

program must correct this anomaly before writing the metadata information to the load-file.

The parsing programs use a series of embedded loops to extract the metadata values from the load file. The entire process uses the element names for each parsing program and uses the 'occurrence' values to find the exact location of a particular metadata value in the load file. The following diagram can represent the process of parsing the load file:

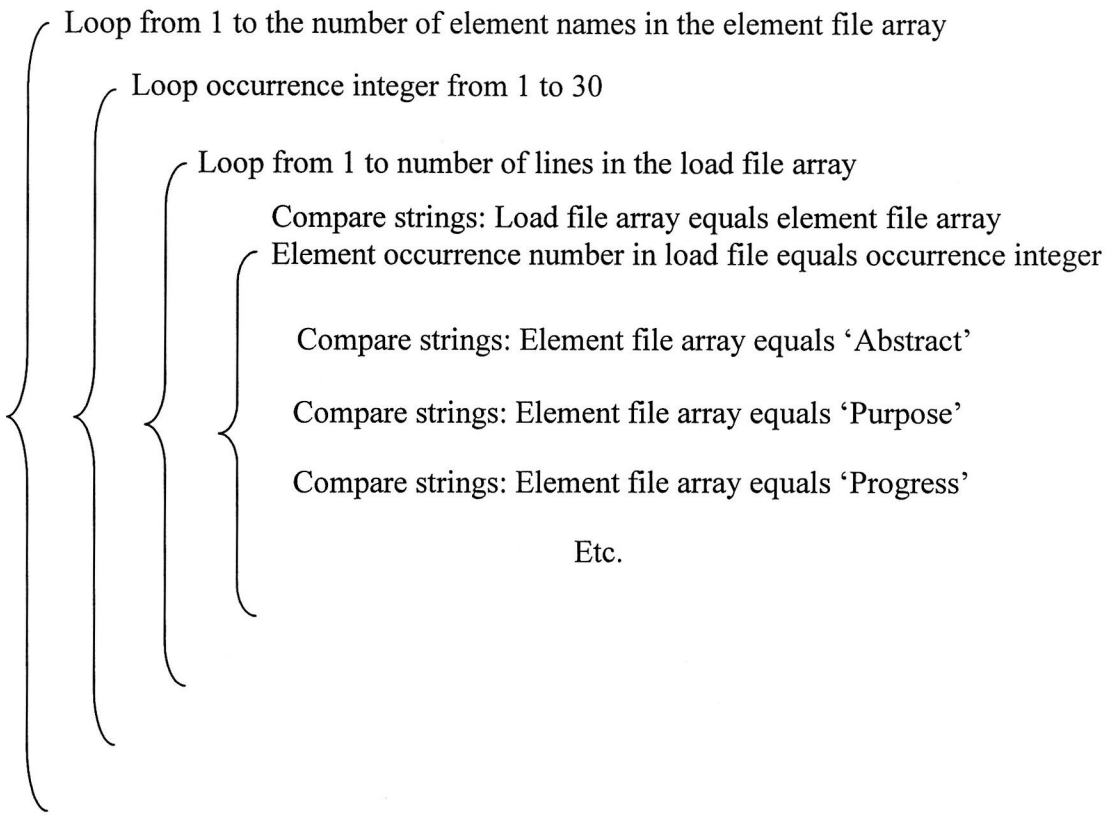


Figure 3.12 The parsing routine for the MLOADER parsing programs. The file is processed using a series of embedded structures. Each level of embedding is used to iterate around one level of multiplicity encountered in the file.

Figure 3.12 shows three imbedded loops. The first loop is incremented from 1 to the maximum number of element names in the element file array. The second loop is incremented from 1 to 30, the maximum number of times a particular element name can occur in the load file. The third loop is incremented from 1 to the maximum number of lines in the load file, to iterate through the load file array. Using the first and second loops as a traversing mechanism, it is possible to locate any metadata

value anywhere in the load file. The fourth bracket of Figure 3.12 is an IF-THEN statement that tests all the possible combinations between the element names and occurrence values. A match is found if the load file array and occurrence number compares to the element name and occurrence integer of the first and second loops, respectively. For example, a single metadata value in the load file is as follows:

```
Abstract
Identification Information
1
Polygons digitized, using the BLM's Automated Digitizing System...
**********
```

The IF-THEN statement first matches the word 'Abstract' with the matching element name of the first loop followed by a match between the occurrence value of 1 with the value of the occurrence integer of the second loop. Inside the IF-THEN statement, the value of the element file array, in this case it is the word 'Abstract', is compared against all the element names for that particular database table. So, for example, the program PII.C, which parses the database table *Identification_Information*, compares all its database columns ('Abstract' being one of them) with the current value of element file array. When a match is found, the metadata shown in the example above is stored in a 2D-array specific for that database column, i.e. *(abstract+ocount). The integer 'ocount' is the value of the second loop of Figure 3.12. It holds the number of the abstract item encountered in the file.

If the parsing of the load file is successful, the result is a series of 2D-arrays specific to a particular metadatabase table. For example, a typical metadata file can have more than one originator of the data set. One or more metadata values associated with the element name 'Originator' can be searched with the procedure described in this section and stored consecutively in the 2D-array, *(originator+ocount). These 2D-arrays, which are declared globally, can then be accessed by the second set of programs designed to insert the metadata into the proper tables and columns of the database. As an error checking procedure, the 2D-arrays are written to an output file for later viewing.

After parsing all the relevant metadata from the load file, the MLOADER program inserts the information into the proper tables and columns of the metadatabase. The database entry programs, which are prefixed with the letter 'd' in Figure 3.5 (dsdi, dci, dii etc.), are controlled by the master.pc program much like their parsing equivalents. Unlike the parsing programs that where executed successively in no particular order, the database entry programs must operate in a specific sequence because of the linkages that exist between database tables.

As described in the first section of the chapter, the metadatabase consists of fourteen tables and five look-up tables each linked in various ways by foreign keys referencing primary keys, which in turn designates the row numbers of each table. The values for one or more columns in a table are restricted by rules known as integrity constraints (ORACLE7™ Server, 1992b). Database references identify the primary or unique key that is referenced by a foreign key in a referential integrity constraint. A referential integrity constraint designates a column or combinations of columns as a foreign key and establishes child-parent relationship between the tables that constitute the foreign key and a specified primary key or referenced key. These constraints are defined in the child table only and the primary or referenced key must already be defined. This is why the order by which database rows are created, updated or deleted is crucial between child and parent tables.

Therefore, to adhere to the referential integrity constraints placed upon the metadatabase by its design, it is necessary to ensure the linkages between tables are maintained as each metadata record is entered into the database. The programs constituting the MLOADER database entry utility operate on a single metadata file forming a single metadata record at any one time. The master.pc program accomplishes the task of database entry by incorporating all the database entry programs into a single loop shown in Figure 3.5. Database entry programs required to run only a single time are executed in the first iteration of this loop. Those programs that run a second, third or fourth time are executed in the second, third and fourth iterations of the loops, respectively. For example, all metadata files consist of just a single row of the *Identification_Information* table in the database. So the DII.PC program is executed only once. However, a particular metadata file can possess more than a single citation, one citation linked by a foreign key to the

Identification_Information table and perhaps two others linked to the Source_Information table. The DCI.PC program in such a situation, needs to execute three times, adding three rows to the *Citation_Information* table. The following Figure shows how a typical run through this database entry loop operates:

Loop 1 – dsdi→dci→dcti→ddi→mri→dii→dsi→ddtp→dbg→ddti→dkt→dkp→dks→dktm

Loop 2 – dci→dcti →ddtp→dbg →dkt→dkp→dks→dktm

Loop 3 – dci →ddtp →dkt→dkp→dks→dktm

Loop 4 – dkt→dkp

.
.
.

Loop 100

Figure 3.13 The program execution sequence of the MLOADER database entry program for a single metadata record. The first loop executes programs that enter metadata from the load-file to the database with 'occurrence' values of 1. The second, third and fourth loops have 'occurrence' values of 2, 3 and 4, respectively.

At this operational stage of the MLOADER program the metadata is contained within the 2D-arrays described earlier. The database entry programs can access their associated arrays and enter its contents into the metadatabase. Figure 3.7 shows the procedure is divided into two parts, one part inserts the actual metadata parsed from the load-file into the database and the other part inserts only the default values ('NULL' for strings and '0' for integers values). Both portions are identical in their operation. The 2D-arrays are simply equated to their equivalent ORACLE external variables in the database entry programs, like so:

```
VARCHAR vcabstract[2000];

strcpy(vcabstract.arr, *(abstract+xyzcount));
vcabstract.len = strlen(vcabstract.arr);
```

The metadata is then inserted into the database for all the columns of the table by its database entry program. So DII.PC uses the following Pro*C SQL statement:

```
EXEC SQL INSERT INTO IDENTIFICATION_INFORMATION
      (IDENTIFICATION_INFORMATION.II_ID,
       IDENTIFICATION_INFORMATION.ABSTRACT,
.
.
.
       IDENTIFICATION_INFORMATION.II_CI_ID,
       IDENTIFICATION_INFORMATION.II_MRI_ID,
       IDENTIFICATION_INFORMATION.II_DI_ID,
       IDENTIFICATION_INFORMATION.II_SDI_ID)
VALUES (:intii_id,
          :vcabstract:indabstract,
.
.
.
          :intii_ci_id,
          :intii_mri_id,
          :intii_di_id,
          :intii_sdi_id);
```

Note that this particular table loaded by the program, *Identification_Information* is the primary table in the metadatabase. It requires the values of four foreign keys referencing rows in the *Citation_Information, Metadata_Reference_Information, Distribution_Information* and *Spatial_Data_Information* tables. The values of these foreign keys must be taken from the primary keys of these four tables. Therefore, the metadata values associated with these four tables must be inserted before the metadata values for the *Identification_Information* table or else the foreign keys, shown in the code above, would have nothing to reference. This point is a further example of the ordered nature by which metadata must be entered into the database; the order is shown in Figure 3.13 with the DSDI, DCI, DDI and DMRI database entry programs executed before the DII program.

Each database entry program uses a Pro*C SQL statement to advance the row number (primary key) of its associated database table. If the associated database table possesses one or more foreign keys, then one or more other SQL statements are used to determine the current value of the primary key of related database tables and equated to the foreign keys. Our example of the DII program uses the command NEXTVAL to increment the sequence table and store the new value representing the last recorded row number of the table *Identification_Information.* The line of code used to do this is:

```
EXEC SQL SELECT IDENTIFICATION_INFORMATION_SEQ.NEXTVAL INTO :intii_id
        FROM DUAL;
```

where intii_id is the primary key value for the *Identification_information* table.

The command CURRVAL returns the current value of the sequence. The current values of the sequences of the database tables *Citation_Information*, *Metadata_Reference_Information*, *Distribution_Information* and *Spatial_Data_Information* tables can be obtained and used as the foreign keys of the *Identification_Information* table. For example,

```
EXEC SQL SELECT CITATION_INFORMATION_SEQ.CURRVAL INTO :intii_ci_id
        FROM DUAL;

EXEC SQL SELECT METADATA_REFERENCE_INFORM_SEQ.CURRVAL INTO :intii_mri_id
        FROM DUAL;

EXEC SQL SELECT DISTRIBUTION_INFORMATION_SEQ.CURRVAL INTO :intii_di_id
        FROM DUAL;

EXEC SQL SELECT SPATIAL_DATA_INFORMATION_SEQ.CURRVAL INTO :intii_sdi_id
        FROM DUAL;
```

where intii_ci_id, intii_mri_id, intii_di_id and intii_sdi_id are the foreign key values to be placed in the *Identification_Information* table.

3.2.3 Design and implementation constraints

The operation of the MPARSER and MLOADER utilities requires the user to have a considerable amount of structural knowledge of the CSDGM metadata standard. The user must be able to identify those metadata sections and elements that require proper delineation (with delimiters) for parsing and database entry purposes. Another constraint is the external variables used in the database entry program have a maximum character limit of 2000 characters. The character limit of ORACLE's VARCHAR2 data type is frequently exceeded by a number of metadata elements such as 'Abstract', 'Purpose', 'Entity and Attribute Overview', 'Resource Description' etc. If records from the metadata elements exceed this limit, they can not be entered into the database, which may result in the MLOADER program to cease execution. Alternative arrangements must be made for metadata that exceeds

this important limitation. The larger metadata elements can be divided into blocks of 2000 characters. Each block can be assigned an external variable and a separate column in the metadatabase. These blocks can be queried individually and assembled into a single value for a particular metadata element.

Other ORACLE7™ datatypes can be used that allow the storage of considerably more information but with certain limitations, which results in substantially less control over the contents. The RAW and LONG RAW datatypes or "BLOBs" used to store raw and unstructured byte-oriented binary data can not be easily searched via conventional algorithmic means. The datatypes are more suited for data associated with graphic files, digitised video and sound. There is also the LONG datatype for storing very long, variable-length character strings. The maximum length in characters of this datatype is 2 Gigabytes, which substantially extends the VARCHAR2 datatype by allowing the storage of very large documents. Even though the LONG datatype can be search via algorithmic means, it suffers from a number of setbacks. A table can only contain a single LONG column and the column can not be indexed. Some GMS metadatabase tables are designed to contain a number of columns containing variable length character strings. The length of the LONG datatype may be restricted by the amount of computer memory while the MLOADER utility is in operation. In addition, LONG columns can not be used with certain clauses (WHERE, GROUP BY, ORDER BY, or CONNECT BY) within the SQL SELECT statement as well as any expression or condition. This precludes its use within the on-line GMS search-engine because these basic SQL commands are fundamental in drawing and presenting information from a relational database (McClannahan, 1996).

3.3 Metadatabase maintenance and management

In addition to an automated facility introduced in the last section for inserting metadata into the metadatabase, GMS also possesses an on-line metadata editor. This utility consists of a series of HTML forms that allow users to enter manually metadata on-line. There is at least one HTML form for each metadatabase table. For each form there is a CGI script that processes the interface values and a Pro*C program that enters the values in the proper columns and tables of the database ensuring the maintenance of referential integrity. Access to any of the tables is through a system requiring the user to specify a database row number, username and password for each table, which is checked against the values stored in the *Identification_Information* table. Database rows can be added or subtracted, viewed and/or changed for any metadatabase table depending on its relationship to the *Identification_Information* table. Users can create new database rows, delete old ones and update current ones at their convenience. Most importantly, their metadata files can be instantly made available to the wider GIS community.

This section contrasts some of the different tools available for on-line metadata editing that operate over the Web. Some of them were briefly described in chapter two. The discussion then moves onto the primary issues concerning metadatabase update and maintenance. Particular emphasis is given to the degree of control that database administrative tools bestow to the data-providers and site-administrators. The last two sub-sections give a detailed design of the GMS on-line metadatabase management and maintenance tools followed by an in-depth description of the their implementation.

3.3.1 The management and maintenance of on-line metadata

The GMS metadatabase and its on-line metadata entry utility provide an effective means to maintain an on-line metadata catalogue by providing those who contribute to the database (i.e. data-providers) with the maximum amount of access to the database without compromising its security. Often, on-line metadata entry utilities must strike a balance in determining the degree of database access between site-administrators and data-providers for the purposes of database update and maintenance. The amount of geospatial data placed on the Web is growing exponentially alongside the rapid adoption of Internet technology among GIS users. The on-line catalogues used to index these geospatial data sets can contain potentially large numbers of metadata records. Administering these records, keeping them updated, ensuring new ones are added and old ones are deleted is a major challenge for the designers of on-line metadata cataloguing sites. Access to the catalogue by a particular data-provider consists of selecting their metadata records for viewing purposes, entering new records into the database, updating their existing records and deleting old ones that have become outdated or redundant.

Though the metadata entry utilities described in the beginning of this section use HTML forms for data entry, the method by which the data is processed and the facilities provided for its update radically differ between systems. The GMS metadata entry utility allows data-providers a considerable amount of access to the tables and columns of the metadatabase. Geospatial data-providers can create new database records over the Web that are secure from other users of the system by a separate username and password verification system for each database record. Metadata records can then be assembled on-line by inserting the metadata into the HTML forms (each associated with a CSDGM section and it elements) provided with the utility. Once the metadata is submitted using these forms, it is entered into the appropriate tables and columns of the relational database. Using the verification system, data-providers can select only the metadata records they have created, update information within each record and delete any of the records that have exceeded their usefulness. Therefore, the GMS metadata entry and modification utility provides

metadata authors with a very flexible system to enter, select, update and delete individual tables, columns or rows of one or more database records.

This level of control is not possible using the BIC Metadata Form because once the metadata is submitted to the server, it is placed in a text file, validated with a parsing program and placed onto the Clearinghouse. If the geospatial data sets associated with the submitted metadata file are altered, then these changes can not be reflected in the metadata file without the data-provider establishing direct contact with the administrator of the metadata catalogue. Such a design increases the cost of maintaining the quality and currentness of the metadata catalogue.

Metamorph (Nebert, 1994) is no different in using discreet text files to store metadata on a centralised server. Though updating facilities were developed for this utility, it tends to rely too heavily on parsing potentially lengthy metadata documents over the Web prior to making any changes to the metadata file. Each time changes are made, the metadata file must be parsed, fed into the Metamorph HTML forms, modified by the user and resubmitted to the metadata catalogue for validation. Retrieving metadata from an RDBMS, as in the case with the GMS metadata entry utility, is far easier than deciphering the contents of a text file using algorithmic parsing.

The GCMD WWW metadata entry utility is somewhat similar to the GMS implementation because both store metadata records within an RDBMS. Metadata records are associated with a unique number (Entry ID), which data-providers use to identify their metadata records. Such a system does not preclude a user from accessing another user's metadata records. This constitutes a considerable security problem for the GCMD database. As mentioned above, the GMS metadata entry utility circumvents this problem through its username/password system designed to associate database records with their authors.

Though GMS is capable of parsing and loading a CSDGM metadata file into an RDBMS, the metadata entry program (described in the present section) provides direct access to the database columns and tables of a metadata record through the SQL SELECT, UPDATE, INSERT and DELETE commands. The use of SQL commands is considerably faster than continually parsing and loading an entire metadata document, particularly when only a successive number of minor changes

are required to the file. GCMD implements such a system by allowing DIF files that already exist on the system to be download to a Web browser where they can be modified and sent back to the system for parsing, validation and database loading. The GCMD metadata entry utility relies entirely on parsing and database loading to insert metadata into the RDBMS. Parsing programs are usually not entirely accurate in extracting metadata from a text file. These programs may operate reasonably well in the GCMD metadata entry utility given the relative simplicity of the DIF metadata standard. However, they are impractical for use within the GMS metadata entry utility because of the size and complexity of CSDGM metadata files.

Therefore, an effective method for site-administrators to maintain their on-line metadata databases containing numerous metadata records is to give data-providers a considerable amount of control in creating and modifying their own metadata records on-line. This sort of control consists of implementing functionality that allows geospatial data-providers to select, enter, update and delete their own metadata records. Validation of the metadata entered into the database can be performed through separate validation software. It can also be performed within the utility itself by examining any modifications to any database record in terms of referential integrity of the database tables or compliance to the CSDGM standard. As a consequence, the maintenance of an on-line metadata database is shared between the site-administrator who oversees the operation of system software and data-providers who keep their metadata records up-to-date.

3.3.2 Software design

One of the ways of giving users effective access to the GMS metadata editing utilities is by providing an 'index page' of HTML database access forms. Each line of the index directs the user to a given metadata entry/modification utility. The on-line editing utility for GMS provides a separate HTML form for each section of the CSDGM metadata standard. The utility consist of fifteen separate forms that are divided into six groups, which include:

- a separate form for *Identification Information,*

- forms for citation, time/date, graphic and keyword information,
- forms for source, source citation and source date/time information,
- a form for spatial data information,
- forms for distribution, date/time, contact distributor and transfer information,
- forms for metadata reference and metadata contact information.

The first form in the proceeding list allows data providers to create on-line metadata records consisting of a bare minimum of information. The forms that follow can be completed in stages or as the information about the data sets becomes available by submitting each form separately to the server to gradually build an entire metadata record.

The 'index page' of the on-line metadata editing utility is shown in Figure 3.14. The form pages link back to the index page providing adequate navigation throughout the utility.

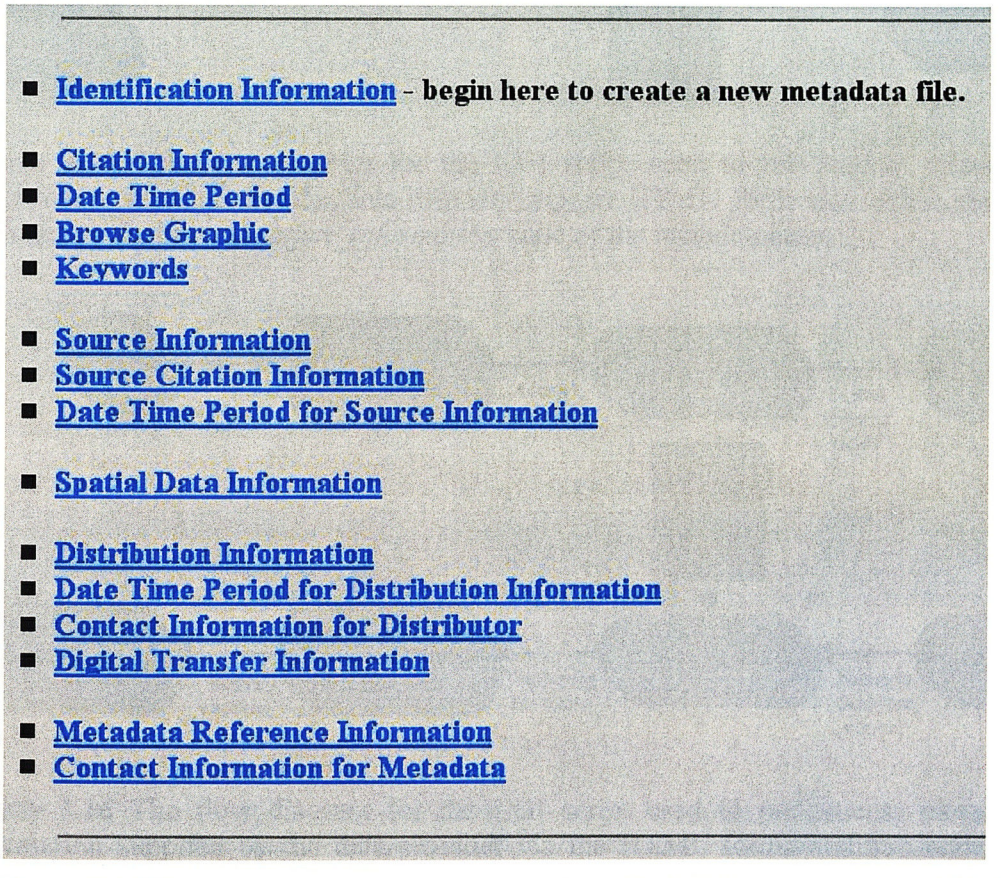


Figure 3.14 The on-line metadata editing utility. Each link connects to an HTML form providing database entry and modification facilities for the named table. New metadata records are created using the Identification_ Information form.

The CGI script used for the *identification_information* HTML form is somewhat different to the other scripts and therefore requires a separate flow diagram to portray its design (Figure, 3.15). Since the CGI scripts for each of the on-line metadata editing forms are similar in design, a single flow diagram (Figure, 3.16) is enough to represent their most common functions.

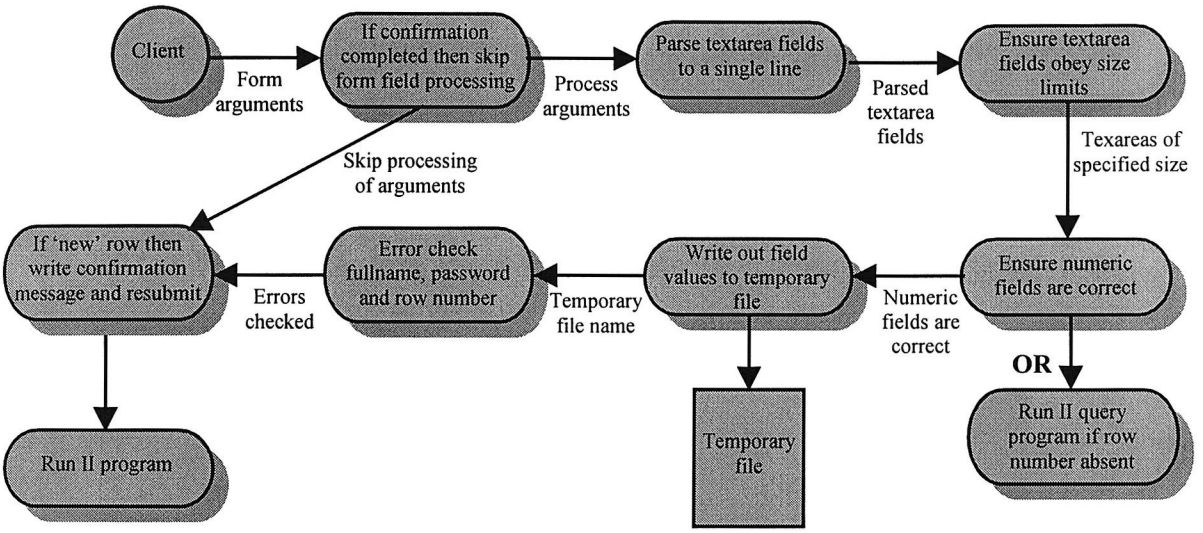


Figure 3.15 The flow diagram for the CGI script used to pre-process metadata information supplied by the data-provider via an HTML form and subsequently entered into the *Identification_Information* table of the metadatabase.

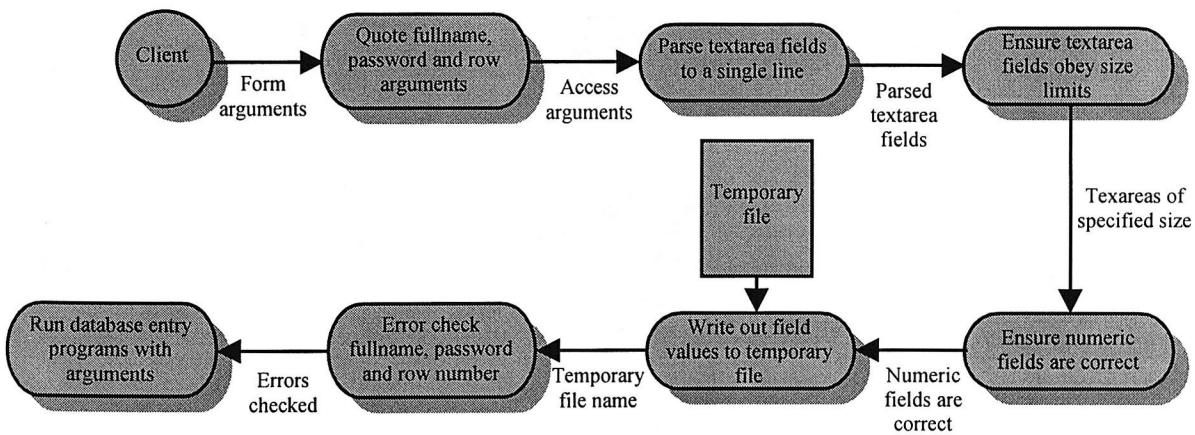


Figure 3.16 The flow diagram for the CGI script used to pre-process metadata information supplied by the data-provider via the HTML forms and subsequently entered into the various tables (excluding *Identification_Information*) of the metadatabase.

The on-line metadata editing utility is divided into three parts, the user-interface consisting of HTML forms (Figure 3.14), the associated CGI scripts (Figure 3.15 and Figure 3.16) and the database entry/modification programs. Below is Figure 3.17 showing the third part of this utility. The database entry/modification programs are developed in Pro*C. Each program is divided into three sections (diagram A, B and C below) showing the utility's INSERT, UPDATE and SELECT operations, respectively. There is one program for every metadatabase table shown in Figure 3.14. The operation of each of these programs slightly differs due to the individual characteristics of their respective database tables. In summary, the utility described in this section consists of software capable of performing four SQL operations, INSERT, UPDATE, SELECT and DELETE (not shown here) on each table of the GMS metadatabase.

Diagram A

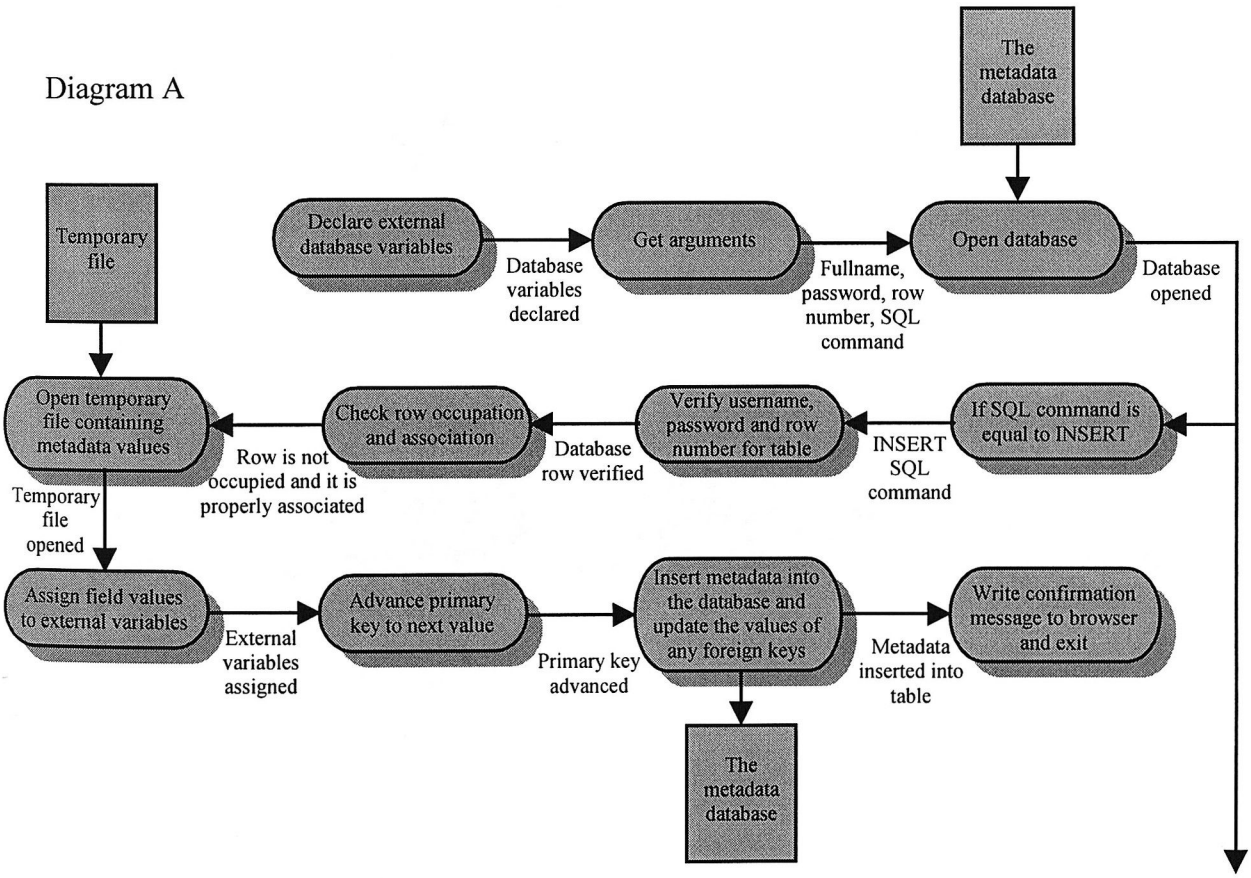


Diagram B

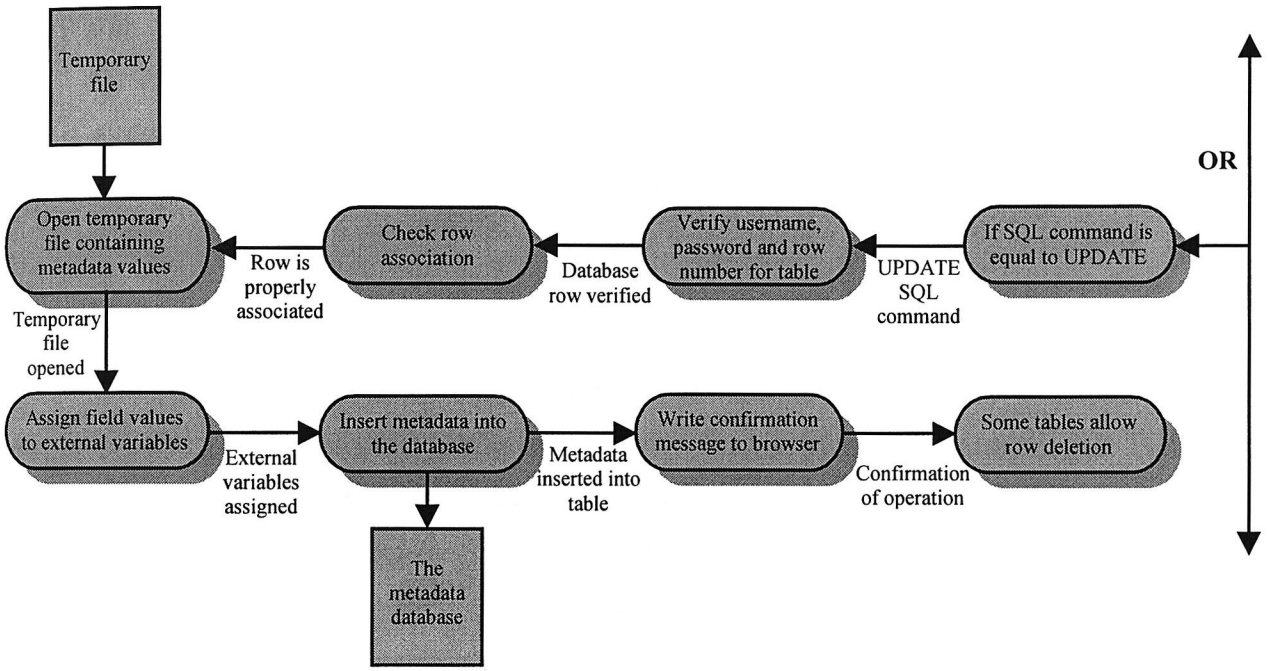


Diagram C

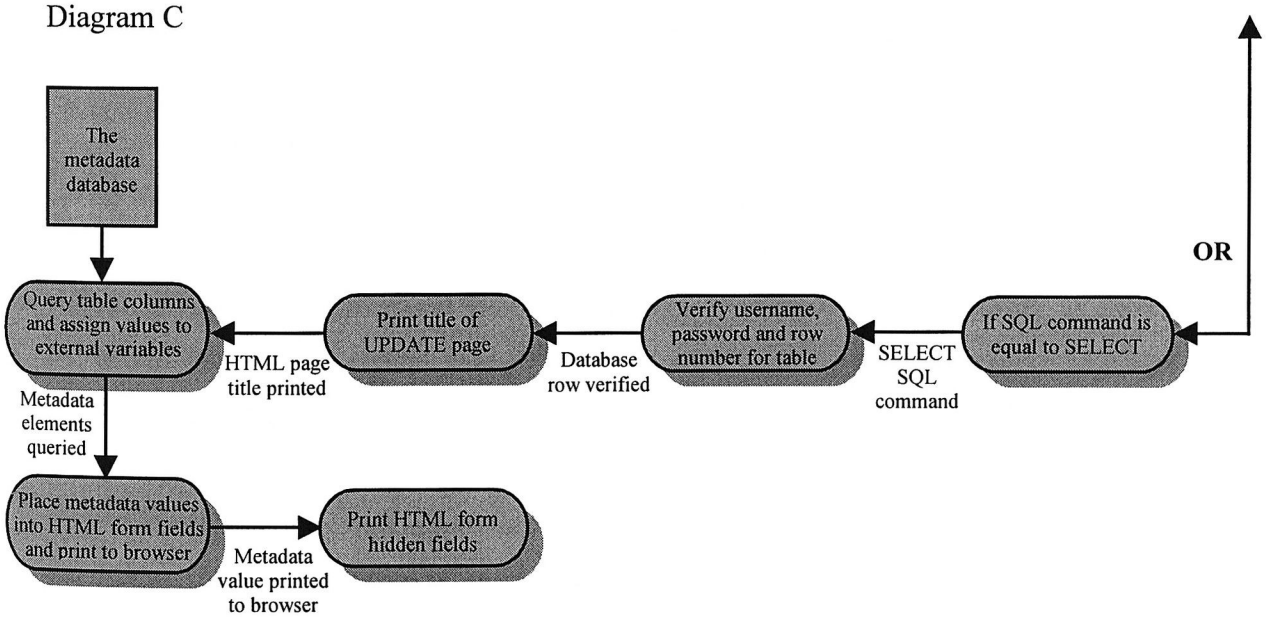


Figure 3.17 A generalised design of all the database entry/modification programs. There is one program for each metadata database table. Each program is capable of inserting (Diagram A), updating (Diagram B) and selecting (Diagram C) metadata values in to and from database tables.

3.3.3 Software implementation

The following sub-section presents a detailed description of the implementation of the on-line metadata maintenance and management utility or simple metadata editor. The commentary focuses on the user-interface, the cgi scripts and the database entry programs associated with the utility. The aim of this section is to give the reader a useful understanding of the working of this rather large on-line software application.

3.3.3.1 The user-interface

The design of the user-interface is as much a part of the software development process of stand-alone applications as it is to those applications that operation over the Internet. The design process for the actual user-interface as well as its implications towards its eventual utilisation is well documented in references such as Dix et al., (1993), Shneiderman (1992), Bass et al., (1991) and Sommerville, 1996. They provide an in-depth discussion of the design, implementation and evaluation of user-interfaces for very large and convoluted software projects. The user-interface for the on-line metadata entry utility along with other GMS components takes an ad-hoc approach for its design because the objectives of this research are confined only to the functional (as opposed to the aesthetic) aspects of GMS.

The user-interface for the on-line metadata entry utility is designed around each of the metadatabase tables and their relationships. The metadata entry forms, as shown in Figure 3.14, are divided into six sections, which closely reflect the order in which sections and elements occur in a CSDGM metadata file. Notice how forms such as *date_time_period*, *citation_information* and *contact_information* are repeated in different sections of the index page reflecting how different portions of the metadata file store the same information in database tables within different sections. Because of the importance placed upon the *Identification_Information* table within the metadatabase, metadata entry commences with the identification_information[3]

[3] For clarity in this section, the names of metadatabase tables are written with their first letters in upper case while the names of HTML forms of the on-line metadata editing utility are lower case throughout.

metadata entry form when beginning a new metadatabase record. The first few textual fields of this form are shown in Figure 3.18.

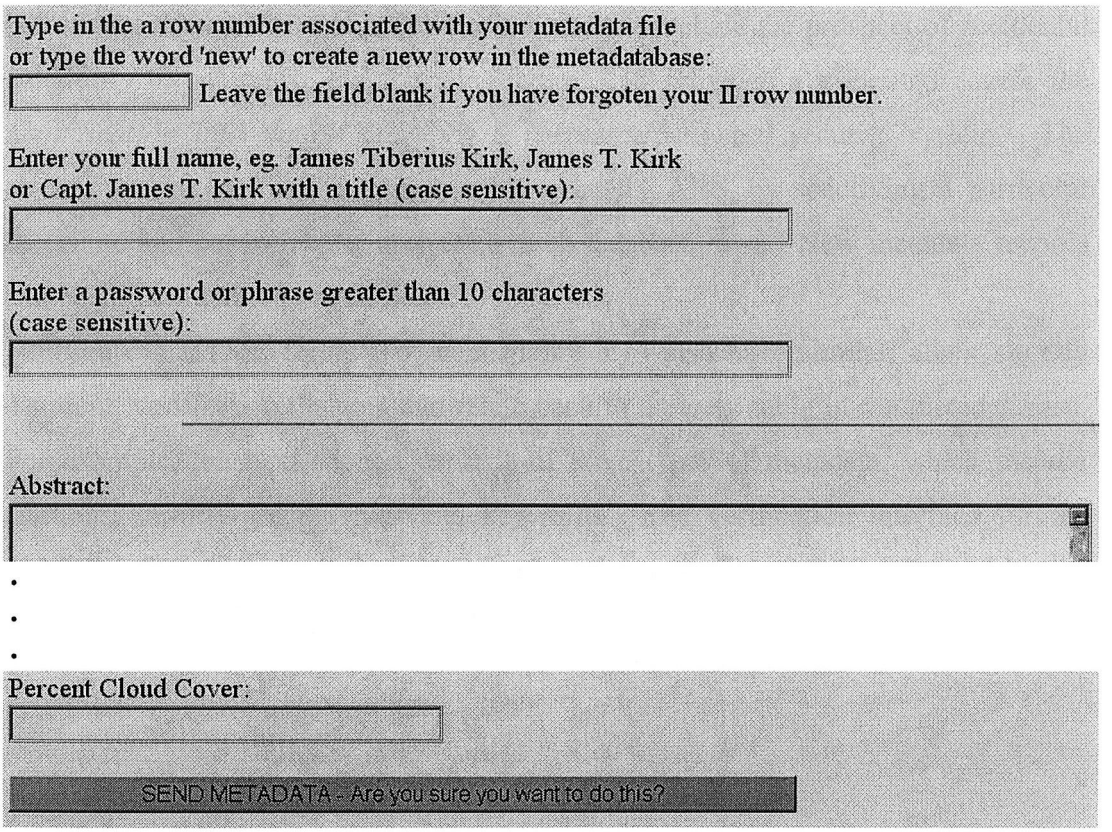


Figure 3.18 The identification_information metadata entry form showing the row number, username, password, some fields and the submit button towards the bottom. The entire form is far too long to be displayed on this page; therefore, only the top and bottom portions are shown.

Users can access a particular database row by specifying a row number, username and password. The software selects (SQL SELECT command) the information from the table and creates a new HTML form on the fly with the information contained within the appropriate element fields, as shown in Figure 3.18. The information contained within the newly created HTML form can be modified and re-submitted by updating (SQL UPDATE command) the information that currently exists in the *Identification_Information* table. Entering the word 'new' in the first field shown in Figure 3.18 can create a new metadatabase row and any information contained within the metadata entry form is inserted (SQL INSERT command) into the appropriate columns of the *Identification_Information* table. If the first field is left blank, the utility is capable of querying the table for all the row numbers that match the

specified username and password. The utility creates a separate HTML form on the fly and lists each row number along with a set of radio buttons, which provides the option to retain or delete the entire metadatabase record. Conformation is required before any database record is deleted in order to preclude the prospect of accidental deletion. In summary, the form in Figure 3.18 provides a means to access the *Identification_Information* table via a username/password security system. The metadata contained in the table can be updated. Also, by specifying a particular username and password the user can search database rows. New metadata records can be created and old ones deleted.

Subsequent HTML forms shown in Figure 3.14 also have similar, albeit slightly different, verification systems that relate back to a parent table in the metadatabase. For example, the next section consists of three types of metadata, which include citations, on-line graphic files and keywords. The verification interface for the citation_information metadata entry form is shown in Figure 3.19 followed by the various element fields associated with this particular database table.

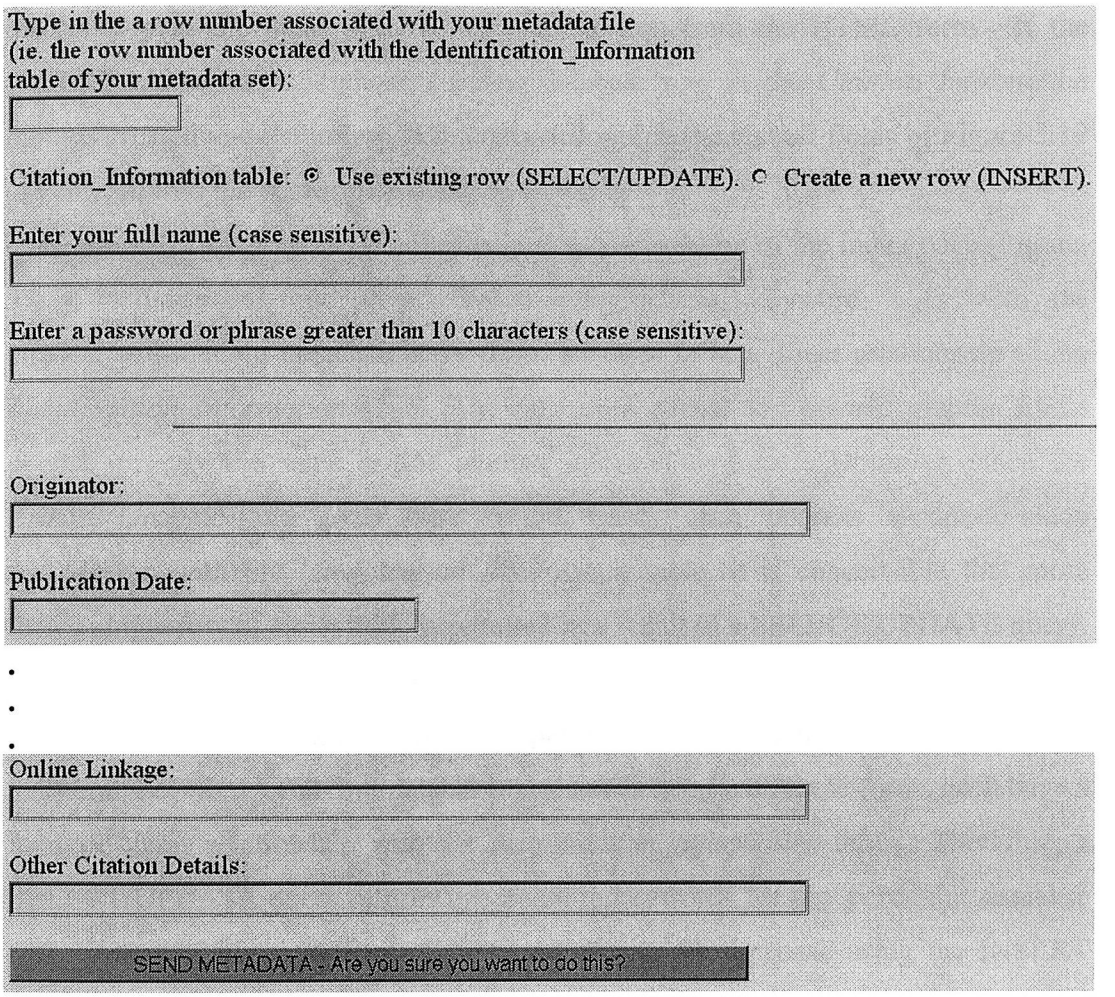


Figure 3.19 The row number, username and password verification for the citation_ information metadata entry form.

The field where the row numbers are entered in Figure 3.19 represents the primary key of the *Identification_Information* table and specifies a metadatabase record. Recall that there can be citation-related information for a geospatial data set and its sources. The row number retrieves only those records of the *Citation_Informaiton* table associated with a one-to-one relationship with the *Identification_Information* table. The type of SQL operation executed depends on the radio button selected. By choosing SELECT/UPDATE, the utility retrieves the rows from the relevant columns of the *Citation_Information* table and inserts the values into another citation_information metadata entry form that is created on-the-fly; a radio button is also provided to delete optionally any of the rows retrieved. The information obtained on-the-fly can be modified and then re-submitted to the

Citation_Information table by an UPDATE operation, which replaces the existing information in the table with the updated version from the HTML form. If the INSERT radio button is chosen, a new database row in the *Citation_Information* table is created and the information contained within the textual fields of Figure 3.19 are entered into the appropriate database columns.

The other HTML forms that comprise the second section of the index page (Figure, 3.14) include date_time_period, browse_graphic and keyword. As with the citation_informaiton metadata entry form, all three have a direct relationship to the *Identification_Information* table. The date_time_period and browse_graphic forms operate exactly the same as the citation_information form. However, since the *Browse_Graphic* and *Date_Time_Period* tables each possess a one-to-many relationship with the *Identification_Information* table, it is conceivable that more than a single row of either table is returned as a result of a SELECT/UPDATE query. The keyword metadata entry form is a single page consisting of four separate forms, which include theme_keyword, place_keyword, stratum_keyword and temporal_keyword. The four database tables associated with these forms each have a many-to-many relationship with the *Identification_Information* table. Therefore, a SELECT/UPDATE query returns one or more keywords for any particular database row. Alternatively, inserting a new keyword into the database using the INSERT SQL command requires the metadata entry utility to determine if the keyword already exists in the database. If a match is found within one of the keyword tables, then only their look-up tables (*Keyword_Theme_Map* etc.) are updated by referencing the keyword with its associated record in the *Identification_Information* table. If the keyword does not exist in the database then it is added to the appropriate keyword table and referenced to a record in the *Identification_Information* table via its look-up table.

The third section on the index page includes metadata entry forms that deal with information related to the source of the metadata. Given that the *Source_Information* table possesses a one-to-many relationship with the *Identification_Information* table, SQL queries from the source_information metadata entry form are performed in the same manner as the citation_information described above. Because of the one-to-one relationship between the *Source_Information* table and the *Citation_Information*

table, each source has a citation and each data set can have one or more sources. The metadata entry form developed to access source related database rows from the *Citation_Information* table is similar to the citation_information form of Figure 3.19 except for the top portion shown below:

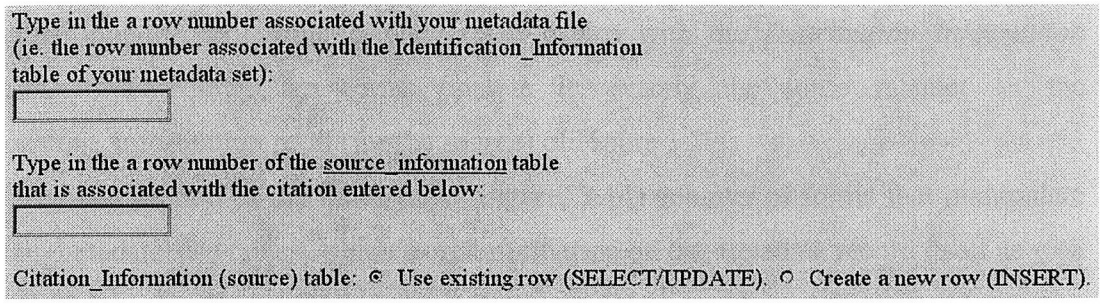


Figure 3.20 The row number fields used to specify source related citation information.

In Figure 3.20 there are fields where the user can specify the primary keys of the *Identification_Information* and *Source_Information* tables. Knowing the two primary keys allows the software to perform a SELECT/UPDATE query on the *Citation_Information* table that retrieves only those rows related to the *Source_Information* table. The first primary key specifies the metadata record while the second value specifies the primary key of the *Source_Information* table. There is also a one-to-many relationship between the *Source_Information* and the *Date_Time_Period* tables. The date_time_period metadata entry form for date information that is related to the source of the metadata record, operates the same as the citation_information form to select, update, delete and insert rows from the *Citation_Information* table.

The spatial_data_information metadata entry form constitutes the fourth section on the index page and is the largest form in the on-line metadata entry utility. This is not surprising since the largest metadatabase table is the *Spatial_Data_Information* table, which possesses a one-to-one relationship with the *Identification_Information* table. Therefore, the spatial_data_information metadata entry form operates exactly the same as the citation_information form shown in Figure 3.19.

The fifth section of the index page (Figure, 3.14) is concerned with data distribution and access. The distribution_information metadata entry form selects, updates, deletes and inserts rows from the *Distribution_Information* table, which is related to

the *Identification_Information* table with a one-to-one relationship. The process, once again, is similar to that of the citation_information form of Figure 3.19. The date_time_period, contact_information and digital_transfer_information metadata entry forms manage database rows for the date, contact and transfer information of the distributor, respectively. The database tables that these forms manipulate possess either one-to-one or one-to-many relationships with the *Distribution_Information* table. Therefore, the forms operate in exactly the same manner as the citation_information form (for the source) of Figure 3.20.

The last section of the index page (Figure, 3.14) consists of forms that manipulate the metadatabase tables, which store information on the metadata record itself as well as the people who compiled the document. The metadata_reference_information metadata entry form manipulates the *Metadata_Reference_Information* table (one-to-one relationship with the *Identification_Information* table) in same manner as the citation_information form of Figure 3.19. Referral related information for those who designed the metadata record is stored in the *Contact_Information* table, which is manipulated using the contact_information metadata entry form (for metadata) in the same manner as the citation_information form of Figure 3.20.

3.3.3.2 The CGI script

The previous sub-section introduced the metadata entry forms used to construct the user-interface of the on-line metadata entry utility. This sub-section concentrates on the CGI scripts used to process the form field values before sending them to the metadata entry programs. Each metadata entry form shown on the index page of Figure 3.14 has its own CGI script except for the Keywords page, which consists of four separate forms handled by their own scripts and processing programs. Hereafter, the CGI scripts are identified with the metadata entry form, which they service, as opposed to their metadatabase tables. Most of the CGI scripts were similarly designed because of shared functionality outlined in Figure 3.16. However, the script outlined in Figure 3.15 is used to process the fields of the identification_information metadata entry form that is somewhat different than the other scripts because of certain added functionality. Such functionality includes, creating new metadatabase records, confirming the creation of each record and

executing a program that associates a given username and password with one or more metadatabase records.

The creation of new metadatabase records is among the most important operation of the on-line metadata entry facility. The system that is devised is easy to use and operates seamlessly with the existing software, which comprises the script and database entry program for the identification_information metadata entry form. The fields of the form are initially filled with metadata destined for the new row of the *Identification_Information* table. After submitting the information with the word 'new' in the row field (Figure, 3.18), the CGI script (Figure, 3.15) processes the form arguments until a point before the execution of the database entry program. At this point, a button is written to the browser prompting the user to confirm the creation of a new metadatabase record (Figure, 3.21).

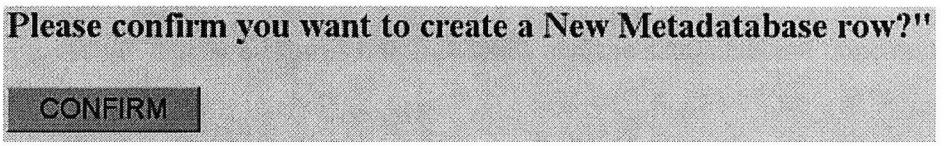


Figure 3.21 The HTML form button used to confirm the creation of a new metadatabase record.

The HTML form button above prevents the accidental creation of metadatabase records, which is what would happen if the identification_information form of Figure 3.18 where designed with a simple radio button to specify new or old rows. The 'confirm' button contains the username and password entered into the identification_information form and its metadata has already been processed and stored in a temporary file (see Figure, 3.15). When the 'confirm' button is pressed, the HTML form activates the same CGI script used to process the field values of the identification_information form. However, the processing of the field values is skipped and the script is advanced to the command line of the database entry program.

The remaining portion of the CGI script for the identification_information form is similar to that of the CGI scripts of the other metadata entry forms. The process of writing the field values of various metadata entry forms to a temporary file is handled differently for each form depending on the type of data processed. Figure

3.15 and 3.16 display all the possibilities that can be encountered but by no means are all these processing steps used for every metadata entry form.

Processing HTML textarea fields is of particular importance to identification_information, spatial_data_information, distribution_information and digital_transfer_information forms. Text entered into a textarea field usually occurs as a series of lines separated by carriage-returns or line-feeds. These lines are then stored in variables within the CGI script and their values are written to the temporary file including the carriage-returns and line-feeds. In this way, text consisting of one or more paragraphs is written on more than a single line in the temporary file. The database entry programs, discussed in the next sub-section, can not read lines of text that occur one after another in the text file. In order for the database entry programs to read effectively the metadata, the script processes the series of lines from the textarea field and joins them into a single line before inserting them into the file. The script removes the carriage-returns and line feeds and the sentences are concatenated into a single long string.

As mentioned with the MLOADER utility in the last section, the largest internal data type used in the metadatabase is VARCHAR2 with a size limit of 2000 characters. The database entry software must respect this size limit or else the SQL commands such as select, update and insert would register an error when executed. None of the HTML text fields used in the metadata entry forms exceed this size limitation because the size of the fields can be regulated using the MAXLENGTH attribute of the HTML INPUT element:

```
<input TYPE="text" NAME="Progress" VALUE="" SIZE=60 MAXLENGTH=255>
```

However, the size of the textarea fields can be limitless. Therefore, after processing the textarea field values to a single string, the CGI script ensures the size of the string does not exceed the maximum allowable size of the metadatabase column for which the string is intended. If a particular string exceeds a maximum size value then the execution of the CGI script is terminated.

Another common internal data type used in the metadatabase is 'number' for storing fixed-size or floating-point numbers. Database columns defined on the

number data type accept only numeric characters. Inserting alpha characters in these columns causes errors. The CGI script ensures that form fields requiring numeric data are in fact numeric values. If incorrect data is entered in numeric form fields then the program prompts the user to correct the error and then exits.

After writing the field values from the various metadata entry forms to the temporary file, the CGI script checks if the user supplied a username and password and ensures the size of these values is a minimum of 10 characters. The field that contains the row number is also examined to ensure a proper row number is entered. All three values, username, password and row number are supplied to their respective database entry programs via the command line.

3.3.3.3 The database entry programs

This sub-section provides a detailed description of the database entry programs, which provide the user with an effective means of storing and managing metadata within their own database records. Each metadata entry form of the on-line metadata entry utility has its own database entry program, which allows database records to be selected, updated, inserted and deleted. The overall structure of each of the program is similar and a generalised flow diagram is provided (Figure, 3.17) that covers the most common steps.

Due to the differing relational integrity constraints that exist on the various tables, each of their respective database entry programs handle inserting and deleting operations according to the sort of relationships that are present. For example, the *Citation_Information* table has a one-to-one relationship with the *Identification_Information* table. If a citation record is already associated with a particular metadata record, then an INSERT SQL operation can not work and the database entry program for the citation_information metadata entry form prompts users to use the UPDATE SQL command instead. The four keyword tables each have a many-to-many relationship with the *Identification_Information* table. New keywords inserted into these tables must first be checked to see if they already exist even if they are related to another metadata record. If existence is confirmed, then only the look-up table needs revision by relating the existing keyword to the new metadata record. See Table 3.1 for a summary.

Table 3.1 Inserting keywords into the keyword tables.

Insert operation on keyword table	Keyword table	Look-up table	Metadata record
Keyword exists	Do not add keyword	Link existing keyword to record	Record is linked to old keyword
Keyword does not exist	Add keyword	Link new keyword to record	Record is linked to new keyword

The same process must occur if a particular keyword is updated or changed to something else. If the existing keyword is related to another metadata record then it can not be deleted from the table; if there is no relation with any metadata record then it must be deleted before the updated keyword is added. If the updated keyword already exists in the keyword table, then only the look-up table is revised by adding a new record. If the updated keyword does not exist in the table then it is added and referenced to the new metadata record via the look-up table. See Table 3.2 for a summary.

Table 3.2 Updating keywords from the keyword tables.

Update operation on keyword table	Keyword table
Updated keyword exists, old keyword is linked to other record(s)	Do not add updated keyword, do not delete old keyword
Updated keyword exists, old keyword is not linked to other record(s)	Do not add updated keyword, delete old keyword
Updated keyword does not exist, old keyword is linked to other record(s)	Add updated keyword, do not delete old keyword
Updated keyword does not exists, old keyword is not linked to other record(s)	Add updated keyword, delete old keyword

The preceding tables demonstrate some of complexity inherent in maintaining the referential integrity after conducting simple database operations. Providing software with the sort of flexibility required for entering, editing and deleting metadata over the Web must account for the possible changes to the linkages between database tables and rows.

Each of the database entry programs contain three separate components shown in Figure 3.17, for inserting, updating and selecting rows from their respective tables. The type of operation performed is determined by a value sent from the CGI script via the program's command line. In addition to this so-called SQL value, the fullname, password and row number values entered by the user are also sent through the command line to verify access for a particular metadata record. Verification

occurs before performing any of the SQL operations. The fullname, password and access number columns are queried from the record in the *Identification_Information* table whose primary key (II_ID) is equivalent to the row number supplied by the user.

```
select ii_id,
       access_number,
       fullname,
       password
 from identification_information
 where ii_id = :row_number;
```

If the fullname and password queried from the database record are equal to the ones supplied by the user, the SQL operation is allowed to proceed. Otherwise, an error message is sent to the browser before closing the database and exiting the database entry program. The access numbers present in every metadatabase table provide the database administrator the means to block access to specific rows. If the access number queried above is equal to zero, then another error message is sent to the user's browser denying access to the particular database row before exiting the program. The access number feature provides an added level of security for individual metadata records in case the fullname and password are compromised.

Given the complexity associated with inserting and updating metadata from certain metadatabase tables (i.e. the keyword example above), it is necessary to determine if particular rows in certain tables are occupied or associated with other database tables prior to performing SQL operations. The database entry programs for certain tables must perform certain checks before inserting or updating values into their rows (Figure, 3.17). For example, because of the one-to-one relationship between the *Citation_Information* and *Identification_Information* tables, metadata records can only be inserted once within the former table using the INSERT command. Multiple records within the *Citation_Information* table can not be associated with more than one metadata record from the *Identification_Information* table; otherwise the referential constraints of the database are compromised. In this case, changing a record in the *Citation_Information* table must be performed using the UPDATE command.

When the citation_information metadata entry form is used to insert a record into the *Citation_Information* table that is linked to a particular metadata record, the database entry program for this table must check if the record is already occupied with metadata. The following query check for row occupation:

```
select citation_information.ci_id
    from citation_information, identification_information
    where citation_information.ci_id = identification_information.ii_ci_id
    and identification_information.ii_id = :metadata_record_number;
```

The query fetches the primary key of the *Citation_Information* table, which is related to the metadata record number entered into the citation_information form by the user. If no records are queried, then the INSERT operation proceeds. Otherwise, an error message is printed to the browser prompting the user that the row is occupied and that the UPDATE command should be used to change the information in the table. The database entry program for tables such as Spatial_Data_Information, Distribution_Information, and Metadata_Reference_Information perform similar checks before executing the INSERT operation.

The example of updating the keyword tables described in Table 3.2 above demonstrates that some of the database entry programs must check for additional references for records that undergo UPDATE and DELETE operations. Because of the many-to-many relationship between the keyword tables and the *Identification_Information* table, individual keywords must be checked for association with more than one metadata record before they are changed to something else or deleted from the database. For example, when a keyword is updated, the old keyword must be check to see if it is associated with any other metadata records, otherwise it is deleted. The updated keyword must also be checked in the same way; if it already exists in the database then a new row in the look-up table is added; otherwise the updated keyword is inserted into the keyword table. The database entry programs for the keyword metadata entry forms use the look-up tables to determine association among individual keywords and metadata records.

```
select keyword_stratum_map.key_stratum_id, keyword_stratum_map.ii_id
    from keyword_stratum_map
    where keyword_stratum_map.key_stratum_id =
:keyword_stratum_row_number;
```

The sample query above examines the number of rows returned from the look-up table and tests association between the keyword table its look-up table (i.e. the primary key value of the keyword table is equated to the keyword ID number in the look-up table, see Figure 3.1). If more than a single row is returned from this query, then obviously the keyword in question is associated with more than one metadata record.

For both the INSERT and UPDATE operations, the metadata is retrieved from the temporary files and each line in these files contains the metadata values of a particular metadata element. The database entry programs open the files in read mode, read the contents one line at a time and assign the values of each line to Pro*C external variables that correspond with their respective metadata elements. Using temporary files in this fashion to store metadata values before database entry is a logical choice since the quantity of information contained therein is far too much to bring in via the command line interface.

If the operation being performed on the database is INSERT, then a new row is added to the particular table using its associated sequence table. The NEXTVAL operator increments the sequence by one and returns the new value (next consecutive primary key value) of the database table. The metadata is then inserted into the proper columns of the table once the next value of the primary key has been established. The addition of a new record to a database table means that the foreign keys of the tables that reference the record must also be updated after the INSERT operation. For example, the foreign key in the *Identification_Information* table that references new records inserted into the *Citation_Information* table must be updated by adding the primary key value of the new record to the associated foreign key.

```
update identification_information
        set  identification_information.II_foreign_key = :CI_primary_key
        where identification_information.II_id = :metadata_record_number;
```

Here the value of the foreign key in the *Identification_Information* table is set to the value of the primary key of the *Citation_Information* table associated with the metadata record number. The UPDATE operation does not need to undergo the process of finding a next value of the primary key because the operation is performed on an existing record. The old metadata in the table is simply replaced with the updated version.

As shown in Figure 3.17B, the UPDATE portion of the database entry program provides radio buttons to delete records from a particular database table. Deleting records in a relational database can have adverse effects on its referential integrity unless proper thought is given to how such an operation affects records and foreign keys from other tables. Both the INSERT and UPDATE portions of the database entry programs provide confirmation messages if the operations are successful.

Selecting records from various tables using the on-line database entry utility is performed in an entirely different fashion than the INSERT and UPDATE operations discussed earlier. The main difference is that rows selected from a particular table are placed in the fields of another metadata entry form created on the fly by the table's database entry program. For example, selecting a row from the *Identification_Information* table begins by specifying a row number, fullname and password in the identification_information metadata entry form. After verifying the fullname and password are associated with the row number, the database entry program selects the specified row from the *Identification_Information* table and places the values of each column into fields of an HTML form created on-the-fly. The HTML form created by the database entry program in response to a SELECT query is identical to the identification_information form and is used to update the columns within the Identification_Information table. Figure 3.22 shows the four operations performed by the database entry programs and the metadata entry forms that initiate the process.

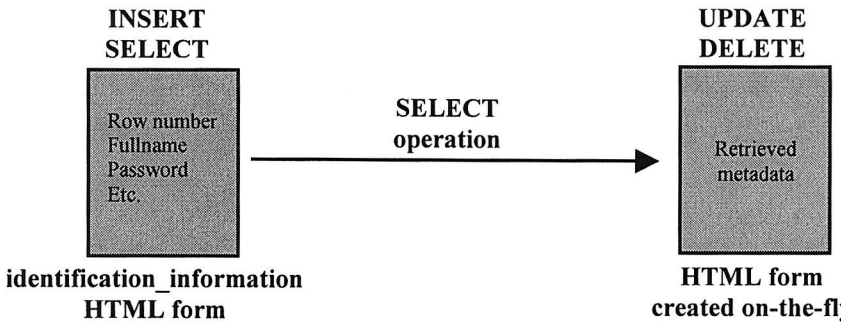


Figure 3.22 Metadata values (row-number, fullname, password, etc.) can be inserted or selected to and from the database. If it is the latter, the retrieved metadata can be updated or deleted from another HTML or created on-the-fly.

The first metadata entry form in Figure 3.22 initiates the SELECT operations that retrieve the metadata from a particular database table and display it within the fields of the second metadata entry form. The second form allows the metadata contained within the fields (each field representing a metadata element) to be updated or deleted.

Lastly, the on-line metadata entry facility possesses tools to determine the metadata record associated with a given fullname and password as well as the means to delete an entire metadata record from the 14 tables and 7 look-up tables in the database. The record querying program is executed from the CGI script associated with the identification_information form and is shown in Figure 3.15. As records are returned from the querying program, a radio button is written alongside each of them allowing the user the choice to either retain or delete the entire metadata record. If the delete button is chosen, then the deleting program is executed removing the entire metadata record from the database after a confirmation message.

3.3.4 Design and implementation constraints

As is the case with the MPARSER and MLOADER utilities described in the previous section, the on-line metadata entry utility requires an in-depth understanding of the CSDGM standard as well as some basic theory on the operation of relational databases. Each of the metadata entry forms has a link to a USGS site

that provides an on-line version of the CSDGM standard for reference purposes. The definitions of metadata sections and elements are provided on-line. HTML anchor elements give the user the ability to navigate throughout the entire document.

3.4 Chapter conclusion

This chapter outlines the design and describes the implementation of several GMS applications that allow data-providers to store and edit considerable amounts of geospatial metadata over the Web. The implementation of the GMS metadatabase is a comprehensive attempt at using a portion of a relatively complex metadata standard for a variety of different on-line applications. Interoperability between system components is enhanced because standardised metadata provides a degree of conformance between data-providers and system developers. GMS stores the metadata within a relational database allowing a common set of terminology for data description, which can be parsed for use within various system components. The process of parsing a metadata file is to separate the information associated with each system element from that of other elements while maintaining some sort of association between the section names, element names and element values. The parsing component of GMS allows site-administrators to maintain the currency, accuracy and quality of metadata that is entered into the database. It is also a means for data-providers to quickly build their own comprehensive metadata catalogues, which are accessible over the Web. Due to the excessive length of metadata files, the on-line metadata editor associated with GMS uses separate HTML forms that allow portions of the metadata record to be submitted separately. The implementation of the most common SQL editing commands allows data-providers a considerable amount of control in creating and modifying their own metadata records on-line.

CHAPTER FOUR

METADATA QUERY & ANAYLYSIS

The previous chapter concentrated on the first three components of GMS, the metadata database, and the utilities used to store and maintain the information contained therein. The next three components (search-engine, data conversion utilities and the Web robot) are described in separate sections of this chapter. They consist of utilities designed to query and process metadata retrieved from the database. The search-engine can query the metadata database using keyword, date and location search-criteria. Among the most unique functions of the search-engine is its ability to query the metadatabase and use the retrieved metadata to operate the on-line conversion utilities and Web robot.

For example, if the metadata retrieved from a query consists of URL addresses, they are examined and the file-types of the on-line resources are identified by the search-engine. In the event the URL addresses reference on-line geospatial data sets (existing anywhere on the Web), then these data sets are downloaded by the conversion utilities and converted to a desired format. This is accomplished by pressing one or more HTML form buttons supplied by the search-engine. Collectively, the data conversion utilities use a common data format to inter-convert between various supported geospatial data formats. This is called a data-switchyard. Suppose the URL addresses mentioned above reference Web pages. The addresses of the HTML documents are fed to a Web robot by the search-engine. The robot visits these Web pages and automatically seeks out geospatial data from remote sites and indexes their hypertext links within a file or database. With the expanding size of the Internet, indexing and cataloguing on-line geospatial data or metadata is increasingly being consigned to automated software.

As with the case of the GMS components described in Chapter 3, the primary method of evaluation of these three GMS components is through on-line use

followed by e-mail feedback. To facilitate this process, the geospatial data conversion utilities, the Web robot and search engine is advertised within comp.infosystems.gis Usenet newsgroup as tools that convert on-line geospatial data and search for metadata within an on-line catalogue as well as throughout the Internet. Select individuals the author deems as important contributors to the human evaluation of these GMS components can be contacted directly and encouraged to use these on-line utilities. To make the evaluation process easier, a detailed on-line explanation of how these utilities operate is provided. Some sample DXF, TIGER/Line and DLG data sets are provided to demonstrate each of the geospatial data conversion utilities. Sample 'keywords' are provided for the GMS search engine to query metadata records stored in the metadata database. A 'canned' demonstration is also provided for the Web robot. The user simply presses a button to begin the program, which supplies it all the necessary arguments to search a pre-specified Web site for geospatial data and metadata. The author responds to any problems, comments or insight received via email from the on-line users and the results of the transactions are documented in the evaluation and discussion (Chapter 5) part of this thesis.

4.1 An integrated WWW search-engine

The Geospatial Metadata Server is a large collection of software components designed to work together in a unified system of information query and exchange. The tool used to accomplish this task constitutes one of the primary constituents of the GMS interface: the search-engine. The functionality present in a metadata search-engine and the tools used in its construction are important considerations when designing information retrieval programs. This section begins by drawing comparisons with various search-engines and the one implemented for GMS in terms of the effectiveness of their user-interfaces and search algorithms.

4.1.1 Tools used to construct metadata search-engines

The growing number of metadata files appearing on the Web facilitates the development of search-engines for data dissemination sites that are capable of performing queries on a large number of metadata elements through a variety of different techniques. At present, these search-engines fall into three broad categories depending on the Web tools used for their construction, which include:

1. Java applet based search-engines,
2. image-map based search-engines,
3. HTML forms based search-engines.

Arguably, one can find search-engines containing elements of one or more of the above categories. Each technology extends the capabilities of constructing search-engines into different directions by providing the ideal tools to develop the functionality for specific tasks of information retrieval. Many of the on-line metadata search and retrieval software examined in this research contain one or more of the above development tools. The categories are not mutually exclusive; they simply represent the range of methods currently available to construct user-interfaces for such systems.

For example, the Digimap interface is an entirely Java applet-based search-engine consisting of a Java applet that can pan and zoom through successive Ordnance Survey map layers by defining rectangular bounding areas or selecting 100 km grid squares. This sort of functionality, which consists of large and active maps, could benefit systems such as the GMS search-engine. Such an interface can be implemented by redesigning the present rather rigid user-interface into one that more easily interacts with the user.

There are two methods by which such a graphical user-interface can be constructed. The first method uses a Java applet that is retrieved by the client each time the GMS metadatabase is accessed. It is executed in an interpretative fashion on the user's Web browser (Anuff, 1996). The second method is through the distribution of a specialised Web client, which is installed on the user's system either consisting of an entirely separate Web client or operating in conjunction with the existing Web browser. Both methods are able to provide panning and zooming facilities over multiple map layers along with the ability to draw circular, rectangular or

quadrilateral search areas using the mouse. Each map layer provides increasingly greater detail of map attributes and topology as the user zooms into successively larger map scales.

However, both the implementation of Java software and the distribution of specialised Web clients have important drawbacks that adversely affect the overall usefulness of the entire system. For instance, problems such as software download, installation and upgrade arise if specialised Web clients are used to enhance existing system functionality. Also, Java technology has serious limitations on its performance particularly when used on portions of the Web where low bandwidth and server load become important considerations when implementing such systems. This is not surprising since the software has to be downloaded from a remote server and run in an interpretative fashion on the local browser. Web software that suffers from such performance restrictions should be used only if no other technology will suffice. Digimap combines its active maps and textual entry forms into a single applet. This is adequate if the number of text entry fields is relatively small in number. However, for search-engines with a larger number of fields, using a Java applet can be rather slow because each time a new form is presented to the user the byte-code must be downloaded from the server, interpreted and displayed on the browser.

The development of an interactive graphical user-interface for the GMS search-engine is a requirement because of the need to provide support for dragging out circular, rectangular and quadrilateral figures for area search requests. Digimap allows rectangular boxes to be drawn onto the interface in order to section an area from an on-line map. It also possesses a small number of form fields. This is in contrast to the GMS implementation, which provides only a point-and-click image-map along with a few HTML text entry forms. Although the search-engine is relatively fast in performing queries; it is rather difficult to use.

Northcutt and Shein (1998) describe a number of different search-engine designs for the Global Change Master Directory. This includes a Web interface, a Z39.50 search interface (both reviewed in sub-section 2.2.2) and a query preview/refinement interface developed in Java. It is possible to construct more than one search-engine, with each based on a different Web technology, for an individual or federated

metadatabase. For example, one of the search-engines designed for the FGDC Entry Point to Geospatial Data Clearinghouse combines both Java applets and HTML forms technologies. The search-engine, not only provides the detail present in large HTML forms, but also has simple and intuitive graphics for selecting attributes or areas from a map. The GMS search-engine could implement such a model, allowing database queries via applets for the more graphic-intensive operations and HTML forms for other metadata related information (date, keyword, citation, etc.).

The GMS search-engine is an HTML form that executes a CGI script, which consists of a number of fields, designed to perform a certain type of query on the database. At present there are three types of queries implemented for the GMS search-engine: date, location and keyword. The location related queries are further divided into three types. The first type consists of a circular search area that is calculated using a geographic co-ordinate and a search radius. The second type consists of a bounding rectangle delineated by a north and west latitude and a south and east longitude. For the last type, the user specifies a quadrilateral search area with four distinct geographical co-ordinates. The query algorithm of the GMS search-engine is constructed entirely using Oracle's Pro*C language. Such an implementation imparts a degree of flexibility in software design that can not be achieved using other database access and development tools such as SQL and interpretative scripts developed in languages such as DCL or PERL. The remainder of the section further clarifies this last point, which focuses on system design and implementation.

4.1.2 Software design

The GMS search-engine consists of a user-interface, a CGI script[1] and a Pro*C algorithm. The user-interface consists of a single HTML page using two HTML forms to supply date, location and keyword search parameters to the CGI script. A flow diagram for the CGI script is shown in Figure 4.1. The script effectively processes the date, location and keyword search parameters before sending them to the search algorithm.

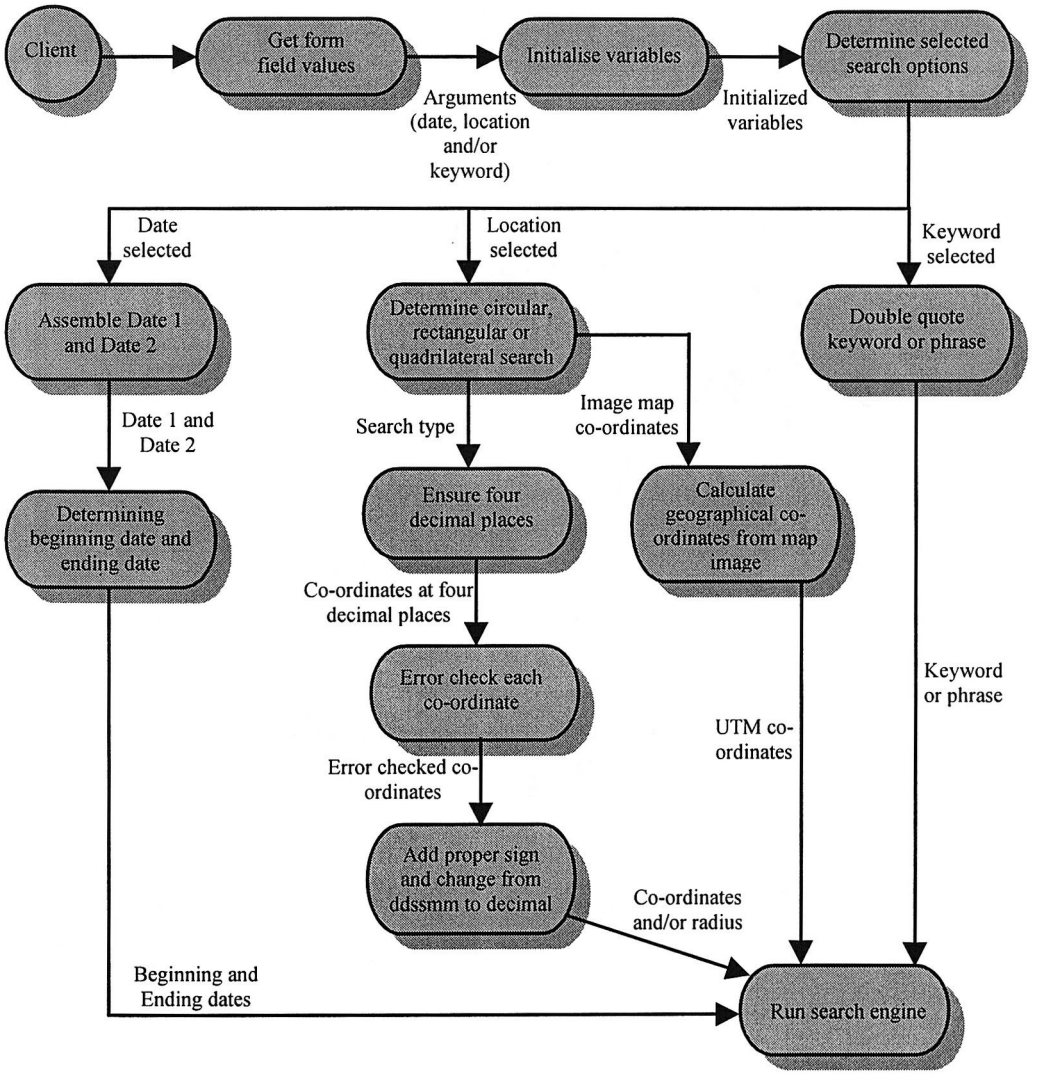


Figure 4.1 The CGI script processes user-specified values from the interface before sending them to the GMS search-engine. This is accomplished within three separate branches of the program that define data, location and keyword search criteria.

[1] Gundavaram (1996) provides an in-dept discussion of all the most relevant aspects of CGI programming.

The search algorithm (Figure 4.2) requires the search parameters (from the user-interface) processed to a specific format before being compared against the information contained in the GMS database. Each type of search query (date, keyword or location) is processed separately by the search algorithm. The database records returned from the query are assembled and the information is then displayed to the client, which includes linkages to the GMS data conversion utilities and other online software.

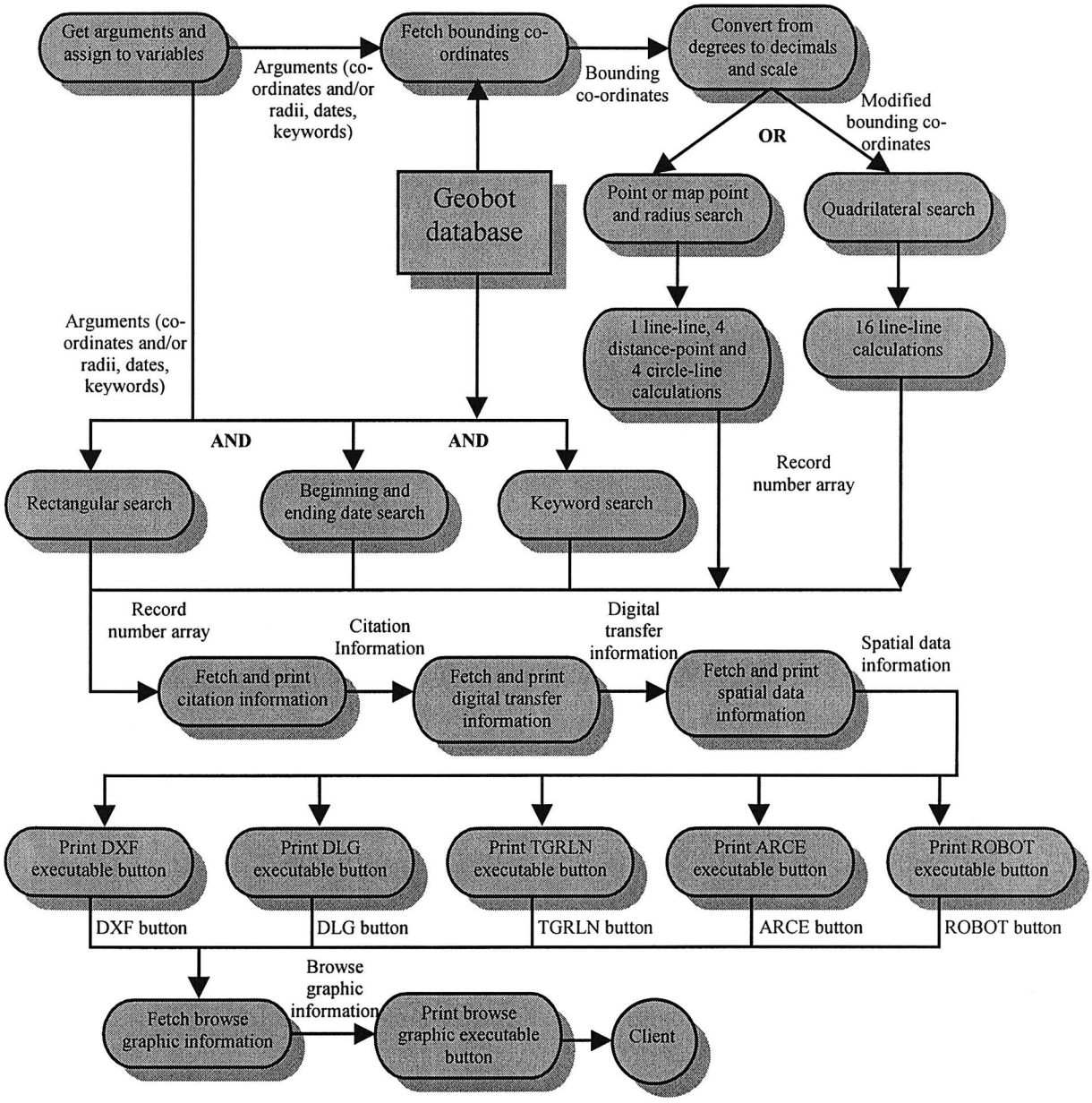


Figure 4.2 The GMS search-engine receives the values from the CGI script and queries the database depending on the search criteria. It prints to the browser, metadata and HTML executable buttons used to run on-line software utilities.

4.1.3 Software implementation

A detailed description of the GMS search-engine is presented in the following three sub-sections. The components of the user-interface are shown in Figure 4.3 followed by the CGI script, which processes the parameters for each type of search pattern before sending them to the search algorithm. The search algorithm retrieves records based on date, keyword or area search patterns, which are constructed through either SQL statements or mathematical geometry. A derivation of the geometry used for the geometric subroutines is given in appendix G.

4.1.3.1 The user-interface

The user-interface for the search-engine was designed to allow the client a fair degree of flexibility in choosing the type of database search mechanism. McLauchlan M. and Gittings B.M. (1995) developed the original version of this interface to query the database of the WWW Earthquake Locator. Since then, the interface has been somewhat modified to allow for greater co-ordinate accuracy and has combined date, location and keyword searches. That latter point is of particular significance because an effective database search mechanism for metadata would have to query on a wide variety of different categories in a concurrent fashion. For example, a database query specifying metadata for a particular time period and location would need to return metadata that refers to data sets whose creation date and bounding co-ordinates fall within the search parameters. Such a mechanism should be able to operate concurrently with any number of the implemented search patterns. The GMS search-engine demonstrates this mechanism using three search patterns as shown in the following figure showing the search-engine's interface.

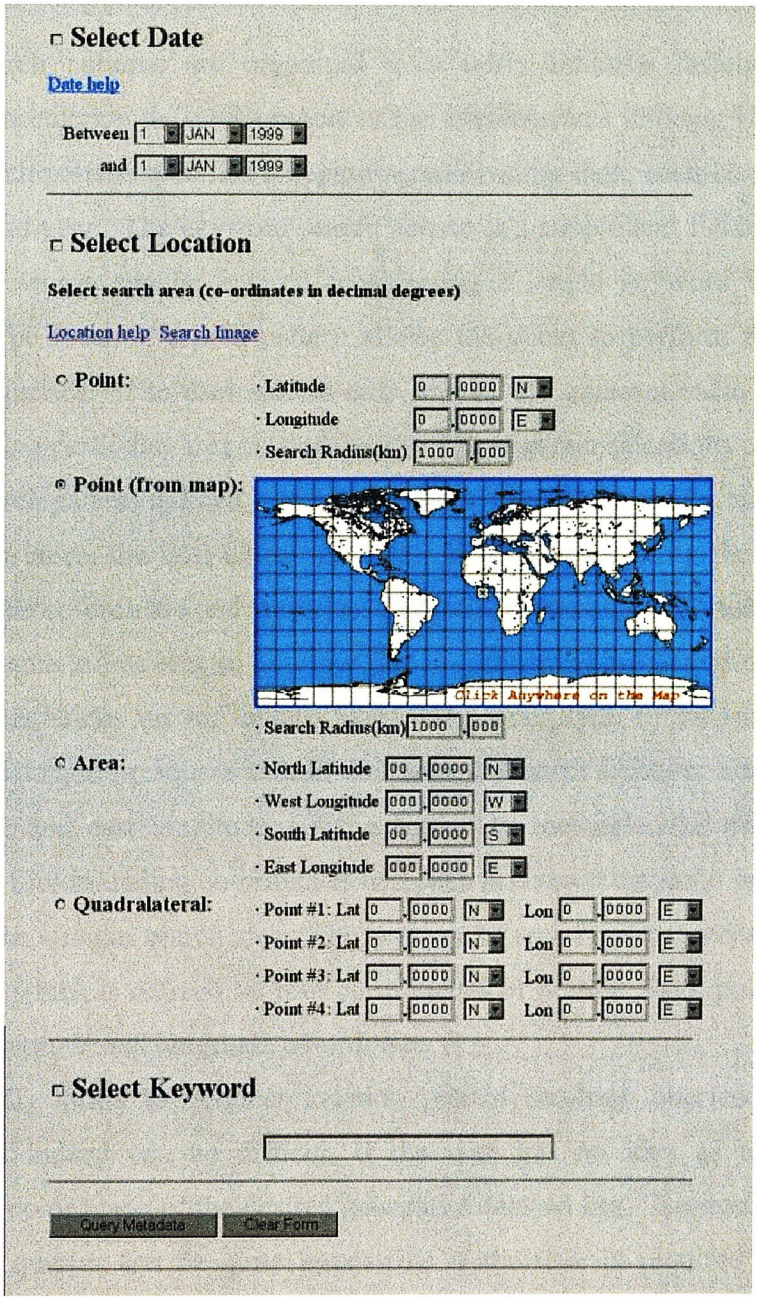


Figure 4.3 The user-interface used to query the metadata database. Queries can be by date, location or keyword search criterion. Search-areas can consist of circles, rectangles or quadrilaterals.

The search patterns (date, location and keyword or phrase) are accessed by selecting the checkboxes adjacent to their names in Figure 4.3. Any combination of these three patterns can be selected. Therefore, to search for metadata using a date and/or

co-ordinate and/or keyword requires the selection of one or more of the three checkboxes.

All the search patterns are organised in a fairly intuitive fashion given the constraints placed upon the development of the user-interface utilising HTML forms. The date search pattern specifies a beginning and ending date, each comprising of a day, month and year. The keyword search pattern is a simple text field where either one or more words can be entered representing a single keyword or phrase of keywords. The location search pattern allows the client to perform four types of area-related queries. The user selects only one of the adjacent radio buttons that corresponds to a particular area-related query. The first two selections correspond to a circular search pattern defined by a single central co-ordinate and a circular radius. In the first instance, the user enters the information by specifying the latitude and longitude of the co-ordinate and the radius length in kilometres. The second instance requires the same radius length, however the co-ordinate is calculated depending on where the user clicks on the image map. The third type of area-related query specifies a rectangle or square that is defined by a north latitude, west longitude, south latitude and east longitude. For a square or rectangle, the parameters are known as the four bounding co-ordinates and each is entered the same way as the co-ordinate for the circular search area. The last type of area-related query specifies any quadrilateral, which is defined by four co-ordinates or points. Each point is defined with its own latitude and longitude co-ordinate.

Using HTML forms to conduct location related database queries is not very effective and indeed can be difficult if the user has no idea as to where the geographical co-ordinate of the desired geospatial data set lies. Because of the same reason, such queries can be quite impossible if the user is required to enter the latitude and longitude of co-ordinates that delineate a rectangle or quadrilateral search area. The only solution to this problem is using a graphical user-interface that allows the drawing of circles, rectangles, squares or quadrilaterals on top of a world map which can be paned or zoomed accordingly. An example of such a graphical user-interface is discussed in a later section.

4.1.3.2 The search-engine CGI script

The CGI script used for the GMS search-engine supplies the parameters for each search pattern (entered through by the user-interface) to the search-engine algorithm after properly formatting each of the numeric and text values. The significance of this step can not be understated because the search parameters can change considerably while travelling from the HTML form to the search-engine algorithm, which is essentially a C program. Unless the search parameters are exactly specified in a format required by the C program, the database query will not work. This sub-section examines some of the problems faced in dealing with parameters associated with the date and co-ordinate search patterns and reveals some fundamental limitations in current HTML and CGI technology.

The CGI script initially determines the type of search method upon receiving the search parameters from the HTML form. The values of each checkbox are compared to determine the search method selected by the user[2]. If the user decides to search for metadata on the basis of date, then the scripting routine responsible for processing date-related information is executed. Such information is stored in the Oracle DBMS as fixed-length, seven-byte values containing the following information (McClanahan, 1996):

Table 4.1 The Date Datatype

Byte #	Value
1	Century
2	Year
3	Month
4	Day
5	Hours
6	Minutes
7	Seconds

So a typical date datatype as represented within the GMS database looks like:

DD-MON-YYYY HH-MI-SS

Where:

[2] See Graham (1996) for a full explanation on the operation of the checkbox element used in HTML forms.

DD represents the day,

MON represents a three letter abbreviated month,

YYYY represents a four-digit year,

HH represents the hours,

MI represents the minutes,

SS represents the second.

Obviously, such temporal accuracy is not needed when representing the date a particular metadata file or geospatial data set was created. However, regardless of how temporal data is presented in a metadata file, the DBMS will represent all date-related information down to the accuracy shown above. If any bytes within the date datatype are omitted, the DBMS will supply them with null values.

To query temporal information from the GMS database, the values supplied to the search-engine must have a similar format to the values contained in the database. Even though the hours, minutes and seconds fields within the date datatype are never used, they still are present along with the rest of the date information in the proper format because the DBMS attaches this information if it is not supplied. Therefore, the task of the CGI script is to assemble 'dates' in the proper format using partial time information as follows:

```
date = day + "-" + month + "-" + year
```

A string, as shown above, is constructed for both the beginning and ending dates. The DCL function, f$cvtime, is used on both dates to return the date values of each string. The two values are then compared to determine the beginning and ending dates; that way, the client need not worry about entering the ending date before the beginning date or vis-versa.

The location routine consists of four sections that process the circle, image map, rectangle and quadrilateral search patterns, respectively. All of these search patterns, except the image map, operate similarly and are defined by geographical co-ordinates, which are composed of degrees, minutes and seconds. The GMS search

algorithm requires these co-ordinates in decimal format to perform the necessary geometric operations (discussed later in this chapter) for locating data sets that fall within the specified search area. The scripting language can not directly handle a degree-to-decimal conversion, so alternative methods were devised to circumvent this problem.

Each co-ordinate shown in the user-interface (Figure 4.3) is composed of three HTML form elements. The first element, as shown in Figure 4.4, is a three-digit field where the degrees are entered. The second element is a four-digit field that handles the minutes and seconds of the geographical co-ordinate. The last field element indicates the direction, which can be either north or south for latitude and east or west for longitude co-ordinates.

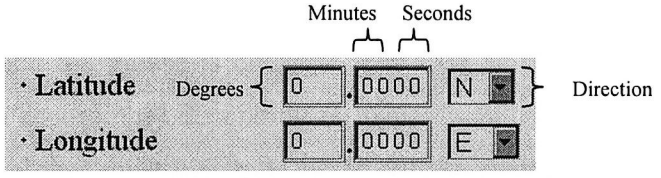


Figure 4.4 A geographical co-ordinate in degrees minutes and seconds.

The CGI script handles each field separately for every co-ordinate of the circle, rectangle and quadrilateral search patterns. There are a number of steps prior to converting the co-ordinates from degrees to decimals. The first step ensures the minutes/seconds field is at four decimal places by running the value through one of the subroutines within the CGI script. The second step is to error-check each field in Figure 4.4 in order to ensure the user enters the proper values. The CGI script must examine the values such that latitudes fall between 0° and 90° while longitudes vary only between 0° and 180° degrees. This check also ensures that the co-ordinate fields on the user-interface do not contain negative values or character strings. Refer to Robinson et al, (1984) for an explanation of the geographic co-ordinate system. The final step uses a subroutine to perform the necessary mathematical calculations that convert each co-ordinate from degrees to decimals. Note that the degrees, minutes and seconds are initially stored in separate DCL variables and then assembled in the following format.

decimal_Latitude = sign + decimal_degree + decimal_minutes/seconds
decimal_Longitude = sign + decimal_degree + decimal_minutes/seconds

The latitude and longitude geographical co-ordinates for the circle, rectangle and quadrilateral search areas are then supplied to the search-engine algorithm as decimal values.

The search pattern using the image map operates in a similar fashion as the circle search pattern described above. However, the geographical co-ordinate acquired from the image map is calculated from the position that the mouse is clicked on a scaled map of the world. The Web browser calculates the x-y co-ordinates at any location on the image map where the mouse is situated. The information is sent to the CGI script when the mouse is clicked. The CGI script then interprets the co-ordinates and acts accordingly depending on which area of the image was selected. Normally, the developer must manually mark out the regions of the image that are associated with a particular action and store this information in a file that can be referred to by the CGI script. However, in the case of the GMS search-engine, the image is a scaled map of the world so the co-ordinate returned to the CGI script will have some relationship to the actual latitude and longitude co-ordinates on the earth. Beginning from the top-left corner of the image map the X-axis varies from 0 to 360 and the Y-axis varies form 0 to 180. Using the following formula, these values are converted to their comparable geographical co-ordinates.

Longitude = X co-ordinate - 180 + 10000
Latitude = (Y co-ordinate - 90) - (2 *(Y co-ordinate - 90)) + 10000

Since longitude varies between 0° and 180°, the first equation simply scales the X co-ordinate between these two values by subtracting by 180. The second equation scales a Y co-ordinate value that ranges from a minimum of 0 to a maximum of 180 to a latitude that must vary between 0° and +90° and 0° and -90°.

The three search patterns, date, location and keyword can query the metadata database individually or simultaneously. The values for each type of search are

processed by the CGI script and supplied to the search-engine algorithm as command line arguments. The next sub-section describes how these values are used to query the metadata database.

4.1.3.3 The search-engine algorithm

The final component of the GMS search-engine examines the values returned for each search parameter and compares them with the records in the database. Records that match the search parameters are printed to the browser. To reiterate, the database records are, in fact, metadata files that give detailed descriptions of one or more on-line geospatial data sets. In addition to printing out these metadata records, the search-engine examines certain database columns or entities of each record and determines one or more appropriate on-line software utilities to process the geospatial data referenced within a particular metadata record. The process where on-line software utilities are executed using arguments derived from metadata records contained in a database is among the main topics in this sub-section.

The search-engine algorithm uses two different methods to compare database records with the user-specified search parameters.

The methods are:

- an SQL based query
- a geometric based query

Both methods use the geographical co-ordinates processed in the CGI script for each type of search pattern. These search parameters initially enter the algorithm as strings but are subsequently converted to decimals using the standard ATOF C language function.

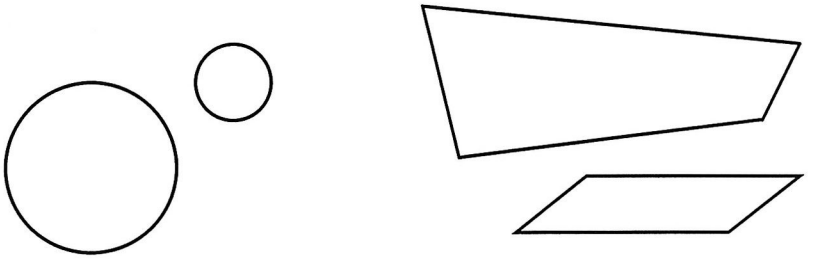
The circle, image map and quadrilateral search patterns use the geometric query method to retrieve and compare geographic co-ordinates from the database. Each metadata record contains information on four bounding co-ordinates (west, east, north and south) that delineate the size of one or more geospatial data sets. The first

step is to query the four bounding co-ordinates from the *Identification_Information* metadatabase table.

```
select identification_information.ii_id,
        west_bounding_coordinate, east_ bounding_coordinate,
        north_ bounding_coordinate, south_bounding_coordinate
    from identification_information;
```

Once again, the co-ordinates retrieved from the database are in degrees/minutes/seconds and must be converted to decimal values for geometric comparison with those co-ordinates supplied by the user. The degrees-to-decimal number conversion are accomplished by a simple C language subroutine within the search-engine algorithm.

As each set of bounding co-ordinates are retrieved from the database and subsequently converted to decimal format, a series of geometric operations are initiated with the database co-ordinates and the search pattern co-ordinates. The circle and image map search patterns are actually the same as far as the database query is concerned. Both are defined using a point co-ordinate and radius (i.e. circular search area). The quadrilateral search pattern is defined as four geographic co-ordinates forming four connected lines (Figure 4.5).



Circle search patterns

Quadrilateral search patterns

Figure 4.5 Some examples of circle and quadrilateral search patterns.

Both of these search pattern queries use C language subroutines implemented in the search-engine algorithm, which perform basic geometric calculations. These calculations are outlined in appendix G. The calculations are relatively simple

operations, which include the intersection between two lines, the intersection between a line and a circle and the distance between two points (Figure 4.6).

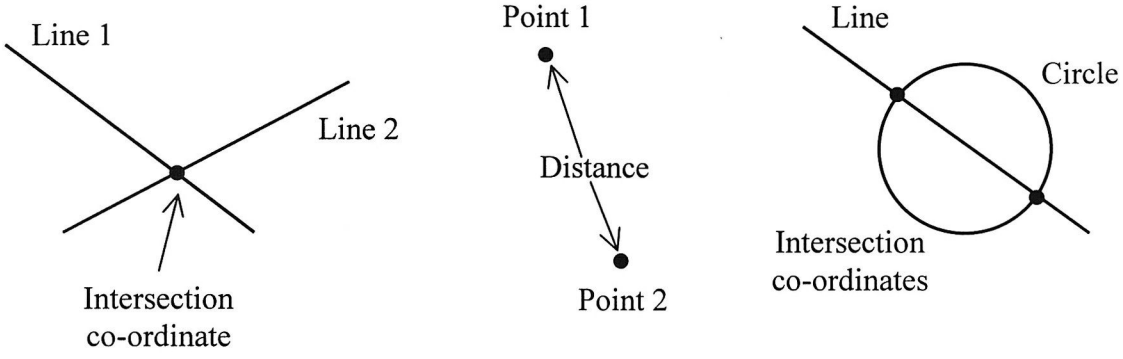


Figure 4.6 Calculation for line-line intersection, circle-line intersection and length between two points (point-distance) for the circle and quadrilateral search patters.

The circle search pattern use all three of the geometric calculations (shown in Figure 4.6) in testing if the user defined circular search area intersects the rectangle delineating the bounding co-ordinates of the metadatabase record. The first geometric calculation uses the line-line subroutine to test if the centre point of the circular search area lies within the confines of the bounding rectangle. This is accomplished by testing orthogonal lines (latitude and longitude) from the centre of the circular search area (top diagram in Figure 4.7). If the intersection point between these lines and the bounding rectangle lie within the circular search area, then there is an intersection between the two geometric figures. The second geometric calculation uses the point-distance subroutine to test if at least one of the four bounding co-ordinates lies within the circular search area. This is accomplished by calculating the distance from each of the four co-ordinates to the circle's centre and then comparing this value to see if any of the resulting values are less than the circle's radius. If the resulting value is less than the circle's radius, then the bounding rectangle of the metadata record and the circular search area intersect (middle diagram in Figure 4.7). The third geometric calculation uses the circle-line subroutine to test if any side of the bounding rectangle intersects the circular search area. All of these tests work in sequence so if an intersection is found then subsequent tests need not be conducted (last diagram in Figure 4.7).

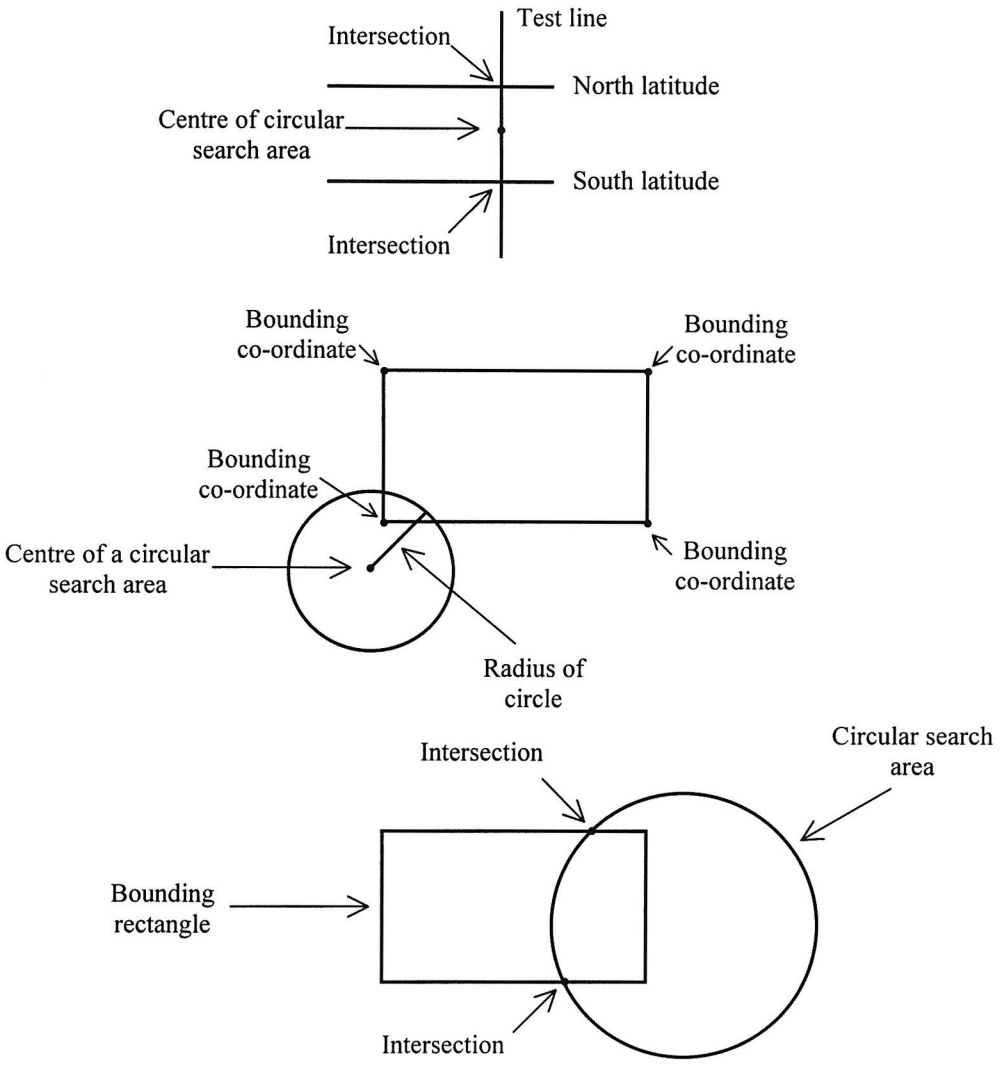


Figure 4.7 Testing sequence for the circle search pattern. See the text for a detailed explanation of each test.

The quadrilateral search pattern performs a somewhat similar sequence of tests to determine the intersection of the search area with any of the bounding rectangles retrieved from each metadata record. There are three eventualities to consider, which include a scenario where the quadrilateral completely encompasses the bounding rectangles, another scenario where the bounding rectangle encompasses the quadrilateral search area, and the final scenario where the side of both geometric shapes overlap each other (Figure 4.8).

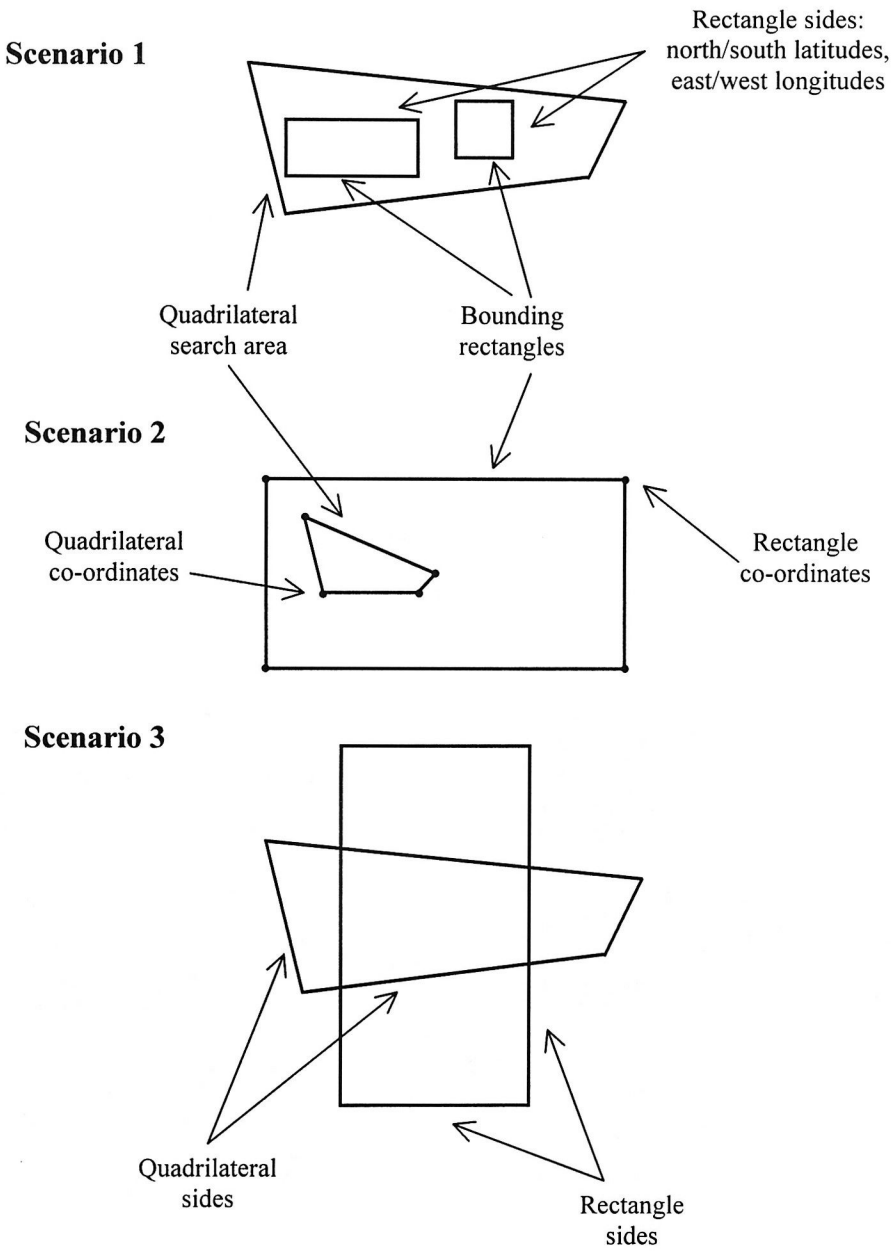


Figure 4.8 Testing sequence for the quadrilateral search pattern. See the text for a detailed explanation of each test.

The first scenario in the above figure uses the line-line subroutine to test if any of the north or south latitudes and the east or west longitude of the bounding rectangle intersect any sides of the quadrilateral search area. The second scenario would test if any of the four quadrilateral co-ordinates fall between the extreme vertices of the north latitude or west longitude (south latitude or east longitude can also be used). The final scenario uses the line-line subroutine to test for intersections between the

sides of the quadrilateral search area or bounding rectangle. As before, all of these tests work in sequence so if a test is successful then subsequent tests need not be conducted.

The second type of query mechanism used in the GMS search-engine is entirely constructed using SQL code. The query comprises of three search types that include a rectangular search pattern similar to the search patterns described earlier. A beginning and ending date used in retrieving metadata records whose referenced data sets are within the specified time period. Also, a keyword or phrase search criterion that is exactly matched with those keywords stored with each metadata record. There are four separate queries for keyword, which include theme, place, stratum and temporal tables. All the database queries are tied together using the UNION set operator which is an SQL function that returns all distinct rows selected by either query (ORACLE7™ Server, 1992a). Figure 4.9 shows a simplified representation of the entire query.

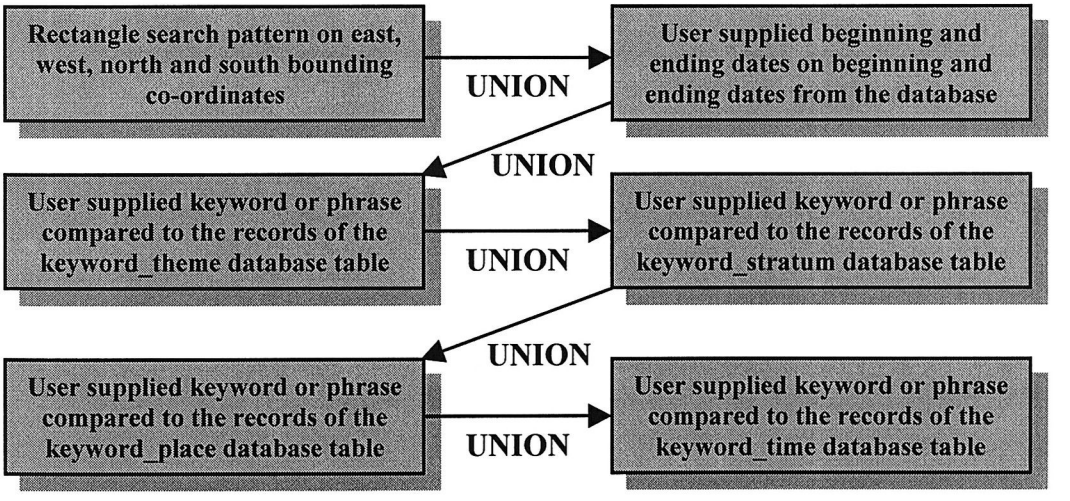


Figure 4.9 The main database query for the GMS search-engine which returns all the distinct rows found from each of the six database queries.

The primary key of the *Identification_Information* table, II_ID, for each database record is queried using both the SQL and geometric search methods. The resulting array of record numbers is used to query and output the information from the metadatabase to the Web browser. The remainder of this section describes how some of the on-line GMS component utilities (data conversion utilities) are designed to respond automatically with information retrieved from the GMS search-engine.

The integration of the conversion utilities and the metadatabase is achieved through the exchange of digital data transfer information between the different components of GMS. Since the information must come from the metadatabase in response to an on-line database query, the GMS search-engine is capable of accessing this information. This is because the database is designed, in part, on the CSDGM standard that has a section specifically describes and references on-line digital data. The table shown in Figure 4.10 constitutes a portion of the entity-relationship model of the GMS metadatabase, which corresponds to a section called the *Digital_Transfer_ Information*. The columns of particular interest to the GMS search-engine and conversion utilities are *Format_Name and Network_Address*.

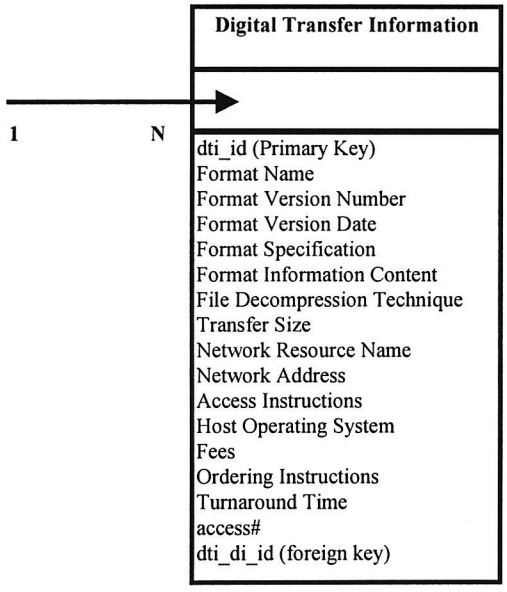


Figure 4.10 The Digital Transfer Information table is reproduced from Figure 3.1. It shows, in particular, the Format_Name and Network_Address metadata elements whose values are drawn from the database and supplied to on-line software utilities.

The columns of particular interest to the search-engine and conversion utilities are Format_Name and Network_Address. Network_Resource_Name is not required because the file name of an on-line geospatial data set can be parsed from the Network_Address. However, File_Decompression_Technique may be useful in the case that a particular compression utility must be identified by the system to decompress the data file. The information in Figure 4.11 along with the *title* and *originator* is outputted for every metadata record returned by the search-engine. In addition, an HTML form button is also provided that allows the user the option to

view the entire metadata record, which includes all fourteen database tables. The reader is referred back to Figure 3.1 in chapter three for a review of the contents of these metadata tables. Figure 4.11 shows a typical query of a metadata record using the GMS search-engine.

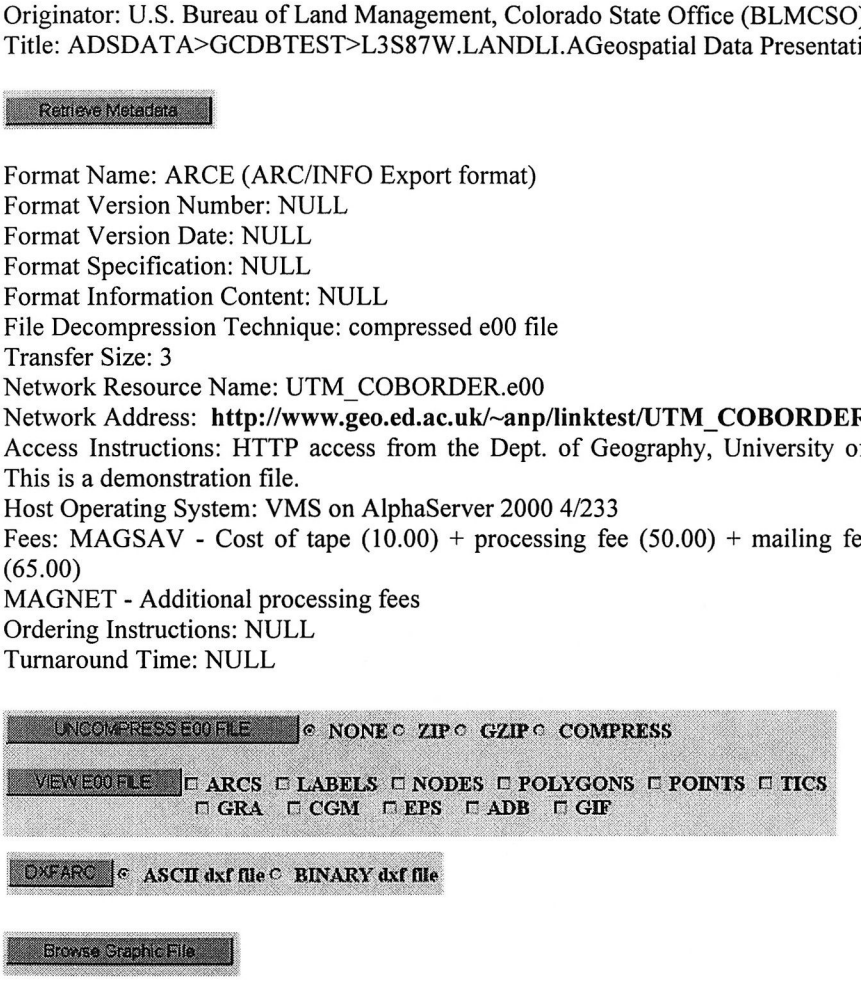


Figure 4.11 A database query of a metadata record using the GMS search-engine.

In addition to providing information for the transfer of on-line geospatial data, a number of metadata files contain hyperlinks to on-line graphic files that illustrate the types of products that can be constructed if one were to use the referenced data. This information is contained within the Browse_Graphic table and includes the file name, description, file type and online linkage (Figure 4.12).

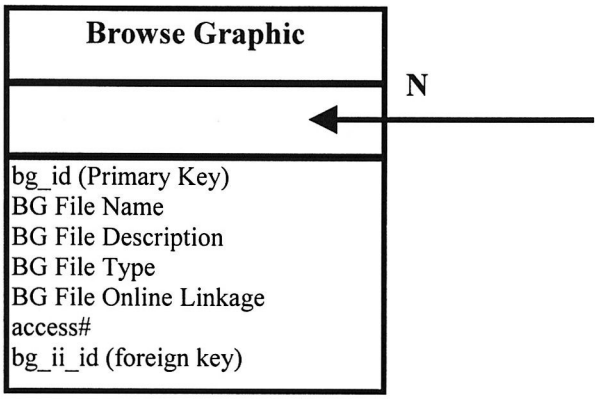


Figure 4.12 The Browse_Graphic table from the GMS metadatabase.

The search-engine is able to detect if this information exists within the Browse_Graphic table, which then creates HTML button(s) that are used to browse on-line graphic files.

```
printf("\n<form ACTION=\"http://www.geo.ed.ac.uk/anpexe/query.com\" METHOD=POST>");
printf("<input TYPE=\"submit\" VALUE=\"Browse Graphic File\">");
printf("<input TYPE=\"hidden\" NAME=\"preview\" VALUE=\"nil\">");
printf("<input TYPE=\"hidden\" NAME=\"browse\" VALUE=\"browse\">");
printf("<input TYPE=\"hidden\" NAME=\"bgfilenm\" VALUE=\"%s\">", vcbgfilenm.arr);
printf("<input TYPE=\"hidden\" NAME=\"bgfiledes\" VALUE=\"%s\">", vcbgfiledes.arr);
printf("<input TYPE=\"hidden\" NAME=\"bgfiletyp\" VALUE=\"%s\">", vcbgfiletyp.arr);
printf("<input TYPE=\"hidden\" NAME=\"bgfileonllink\" VALUE=\"%s\">",
vcbgfileonllink.arr);
printf("</form>");
```

Figure 4.13 An HTML executable 'button' to browse on-line graphic files.

All of the information from the metadatabase table is contained within hidden HTML fields of the 'executable button' and is sent to the CGI script on pressing the HTML button (Figure 4.13). The script then produces an HTML page on-the-fly with an imbedded graphic file.

The same procedure is used for the data conversion utilities. After completing all the queries, extracting all the information from the relevant database tables and reconciling any conflicting metadata records from the location, keyword or date queries, the search-engine uses the queried information to provide linkages to executable programs on the server. The *Format_Name* column of the *Digital_Transfer_Information* table is used to determine those data-conversion utilities that are appropriate to use with the on-line geospatial data (referenced in the

metadatabase as URL addresses). Figure 4.14 shows how the search-engine accomplishes this:

Directing system output to the Web browser:

```
if (!strcasen(format, "DXF", 3))
{
 printf("\n<form ACTION=\".....\" METHOD=POST>");
 .
 .
 .
 printf("</form>");
}
if (!strcasen(format, "DLG", 3)) { HTML executable button }
if (!strcasen(format, "TGRLN", 5)) { HTML executable button }
if (!strcasen(format, "ARCE", 4)) { HTML executable button }
if (!strcasen(format, "ROBOT", 5)) { HTML executable button }
```

Figure 4.14 Sample code from the metadatabase search-engine.

In compliance with the CSDGM standard, DXF, DLG and TGRLN in the column *Format_Name* are standard acronyms indicating that the URL in the column *Network_Address* references data in those proprietary data formats (Federal Geographic Data Committee, 1994). On-line software can be accessed through 'HTML executable buttons' to convert any of these formats to Arc/Info's e00 export format. The e00 export file format is represented by the ARCE designation. Alternatively, 'HTML executable buttons' could be provided in the ARCE IF-THEN statement shown in Figure 4.14 to convert online e00 files to DXF, DLG or TIGER/Line data formats. However, the complexity of these particular conversion utilities prevents this at present.

Within the same statement block (Figure 4.14) the search-engine provides executable buttons that decompresses e00 files (import e00 → export e00 without compression) and an on-line Arc/Info coverage plotting utility.

What would happen if the Network_Address (Figure 4.10) column does not reference a specific geospatial data set but in fact provides a URL address to a site that contains tens, hundreds or even thousand of data sets? The GMS search-engine (through the last IF-THEN statement in Figure 4.14) provides access to a Web Robot just for this eventuality. The robot, called Geobot, scans a site domain HTML page by HTML page for links to geospatial data looking at their file extensions. Geobot is discussed in more detail in section 4.3.

4.1.4 Design and implementation constraints

One of the more obvious constraints in developing the GMS search-engine is using HTML forms to construct the user-interface. Delineating the location-related search types with even simple geometric shapes such as circles, rectangles, quadrilaterals is exceedingly difficult by specifying geographic co-ordinates. Given the accuracy supported by the search-engine, a simple image map of the size shown in the user-interface can only give a very rough approximation of the position of any particular geographic co-ordinate. Multiple image maps linked to one another showing progressively larger scales to simulate the effect of 'zooming' can be constructed, however the co-ordinates must still be entered via an HTML form.

The CSDGM standard is not particularly easy to use in describing geospatial data sets. Some of the acronyms and terminology are overtly complicated, lack expressiveness and flexibility. In reference to the search-engine, terminology provided for the CSDGM element, Format_Name, in the form of three to five letter acronyms that represent a particular data set type are not widely known to the geospatial data-user community. The search-engine depends on the author of the metadata file being able to supply the proper acronyms so that the correct HTML executable buttons are paired with their designated geospatial data sets. Further work on the search-engine could include more comprehensive guidance and help pages that summarise the more critical terminology of the CSDGM standard.

The search engine provides a simple demonstration of accessing the records contained in the metadata database through date, location and keyword search parameters. The search parameters used in this search engine represent only a small number of metadata elements. This is not uncommon among many on-line metadata catalogues because they implement only a small number of metadata elements, used solely as search parameters, within their databases and then link these records to flat files stored locally. Since the GMS metadata database is modelled on part of the CSDGM standards, one can conceivably provide search parameters for any metadata element or attribute in appendix C. All that is required to construct a more

comprehensive search engine is to extend its interface and CGI scripts to accommodate more attributes.

In particular, the search engine can be extended to query on all the citation related information contained in the *Citation_Information* table. Other possibilities include searching for remote-sensing images based on the *Percent Cloud Cover* attribute contained in the *Identification_Information* table. Geospatial data can be searched on the basis of graphic images or representations of the data set, which are documented within the *Browse_Graphic* table of the metadata database. Users can also search the database for particular distributors of geospatial data or the authors of the metadata records themselves.

4.2 Geospatial data-conversion utilities

The discussion so far shows that metadata can be stored, maintained and queried over the Web for the purposes of cataloguing and distributing geospatial data. Up till now, the primary function of metadata within the GMS system is its ability to provide a standardised terminology for describing on-line geospatial data sets. In short, it is a means to search and discover data resources that are of use to the GIS user community. The following section departs from this primary metadata function by examining how such information can be used with software designed to operate over the Web. This sort of functionality is implemented within the GMS search-engine, which was reviewed in the last section. It can furnish metadata information as arguments to three data-conversion utilities, which can subsequently be executed using common HTML interface components. This section outlines the design of each data-conversion utilities followed by an in-depth description of their implementation.

4.2.1 Geospatial data formats and data-conversion

There are a number of standards designed to encode geospatial data for transfer over distributed networks such as the Internet. These data standards address the growing problem of GIS interoperability as a consequence of the incompatibility of vendor-specific data formats and their widespread use by GIS users accustomed to accessing such resources on-line. All of the data formats used within the GIS industry use either the raster or vector data models to represent geographic entities. The data formats used to construct the GMS data-conversion utility use the vector data model. These spatial data standards primarily differ in their amount of object-orientation and the extent of integration or separate representation of spatial and non-spatial characteristics (Healey et al., 1998). However, this is considered a high level of representation for data standards. At the lower or basic level of representation,

features consist of spatial and non-spatial attribute components that are quite similar between different geospatial data standards.

Despite the underlying similarities between the vector spatial data formats, data conversions from one particular format to another and back again must be performed by software designed specifically for that purpose. There are currently no 'meta' data converters capable of translating any spatial data format to and from any other spatial data format. The nearest the GIS industry has come to such a device is by converting from a common data format to any other spatial data format using a separate data converter for each format. By reversing the process and converting a particular data format back to the common data format, any particular data format can be converted to any other data format.

FME is a commercial software product that is developed and marketed by the company known as Safe Software (Safe Software Inc., 1999). It is a software package that uses the Spatial Archive and Interchange Format (SAIF) to inter-convert between a number of spatial data formats. SAIF, the Spatial Data Transfer Standard (SDTS) and the Neutral Transfer Format (NTF) were all created by government-funded standards organisations, thereby freeing their development from proprietary influences. This is in contrast to the proprietary conversion utilities used within GMS. Since Arc/Info version 6 is one of the primary software packages used in implementing GMS, its conversion commands were used because of the relative ease and speed of their implementation within the system. The commands used within the GMS conversion utilities are capable of converting DLG, DXF and TIGER/Line vector data to and from the Arc/Info interchange format (e00). GMS uses the e00 format as its common data format to inter-convert between other spatial data formats in what is known as an on-line data-switch-yard.

Each GMS data-conversion program usually possesses a large number of options reflecting the complexity associated with converting geospatial data to and from Arc/Info coverages. With further development it is possible to simplify considerably the user-interface. The conversion utilities can be re-designed for more general use by removing or rephrasing any overtly technical terminology and reducing the number of available options. Alternatively, a simple and a complex version of the utilities can be maintain to satisfy a wide assortment of data-conversion needs.

Because the conversion algorithms used for the GMS conversion utilities are proprietary, they require proper licensing arrangements with the supplier before accessing them freely over the Web. Given that this project is purely research-orientated, the software developed thereof is for demonstration purposes only. Extending these conversion utilities and providing a more permanent, freely available service to the GIS community requires the integration of freeware conversion utilities currently available from the Web.

The data-conversion utilities introduced in this section are part of the Arc/Info version 6 GIS software package and uses ESRI's e00 interchange-file formats to process and convert geospatial data to and from various other formats (ESRI, 1991). Specifically, the data-conversion component of GMS is design to convert data files of e00 format to and from the Drawing Interchange File (DXF) format from AutoCAD, the Digital Line Graph (DLG) data format and the Topologically Integrated Geographic Encoding and Referencing System (TIGER) format, which are both USGS products. Not surprisingly, the three-geospatial data-conversion utilities that comprise this component of GMS are known as DXF, DLG and TIGER. In all, there are six conversion commands each comprising of several components. The Web-based utilities are named after the data-conversion commands they implement. These are outlined with their arguments in ESRI's help documentation below:

e00 ←→ DXF
ARCDXF <out_dxf_file> {in_line_cover} {in_point_cover} {in_annotation_cover} {decimal_places}
{ASCII | BINARY}
DXFARC <in_dxf_file> <out_cover> {text_width} {attrib_width}

e00 ←→ DLG
DLGARC <STANDARD|OPTIONAL> <in_dlg_file> <out_cover> {out_point_cover}
{NOFIRST|ALL|ATTRIBUTED} {x_shift} {y_shift} {category}
ARCDLG <in_cover> <out_dlg_file> {in_point_cover} {in_projection_file} {x_shift} {y_shift}
{in_header_file} {TRANS | NOTRANS}

e00 ←→ TIGER
TIGERARC <in_tiger_file_prefix> <out_cover> {out_point_cover} {out_landmark_cover}
ARCTIGER <out_tiger_file_prefix> <in_line_cover> {in_point_cover} {in_landmark_cover}

Figure 4.15 Arc/Info data-conversion commands used in GMS.

Each conversion command comprises of several software components. The components include an HTML form that provides a means to execute the conversion

utilities and assign the appropriate arguments to the Arc/Info data-conversion commands, a CGI script to control server-side software, a data-conversion utility and a number of customised software components. In addition to data-conversion, the utilities, ARCDXF, DLFARC and TIGERARC do optional coverage processing over the Web before the resulting coverage is exported to the e00 format for transfer back to the user.

4.2.2 Software design

The DXFARC, DLGARC and TIGERARC conversion utilities are all similarly designed and therefore, require only a single generalised diagram for their representation (Figure 4.16). The conversion utilities translate DXF, DLG and TIGER/line data to Arc/Info's e00 interchange file format. They differ in the design of their AML programs, which are shown in subsequent diagrams. Of particular importance in Figure 4.16 is the implementation of the uploading programs that are used in each of the GMS conversion utilities. The design and description of the HTTP and FTP file uploading utilities is provided in appendix F.

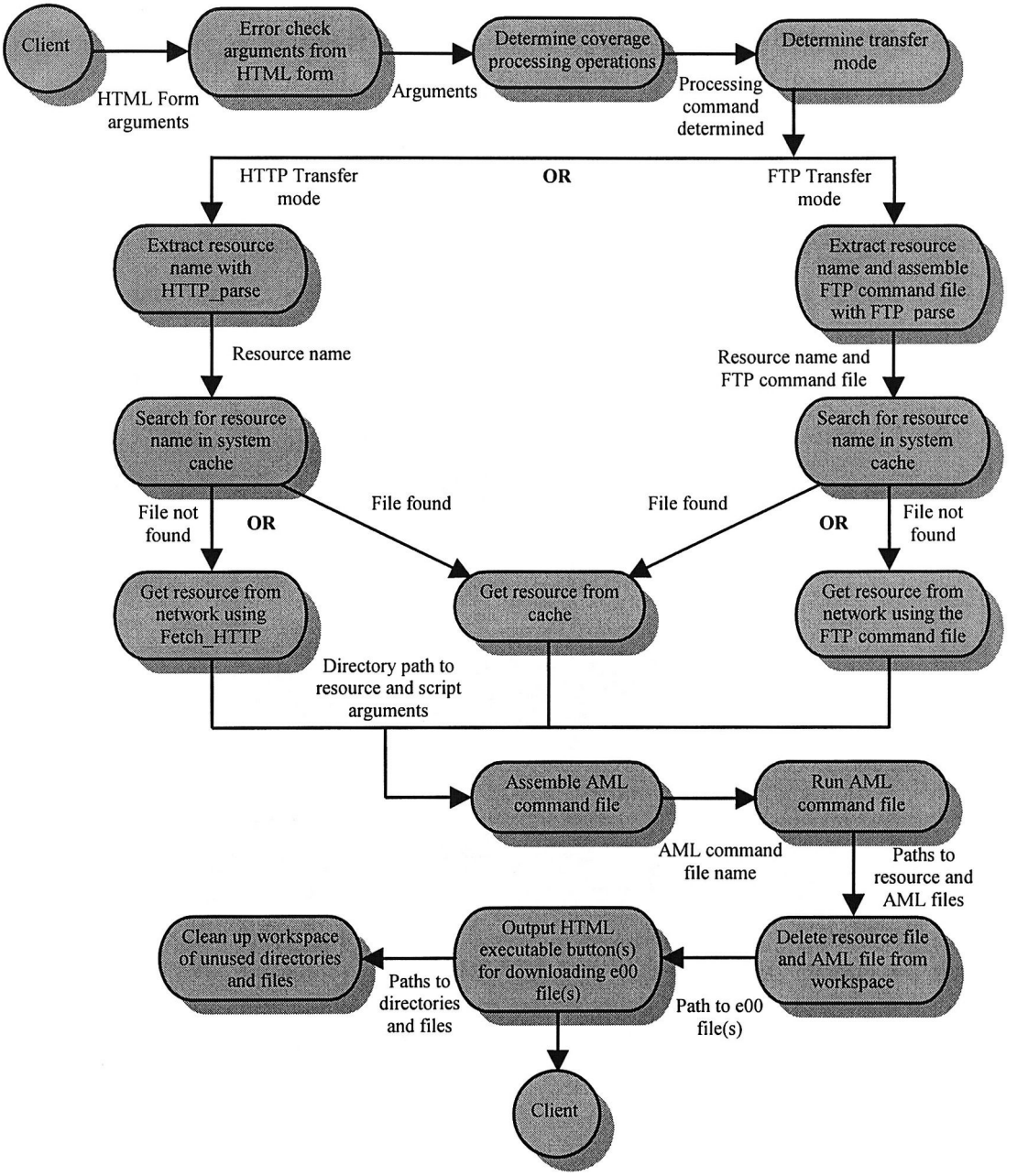


Figure 4.16 Flow diagram for the DXFARC, DLGARC and TIGERARC geospatial data-conversion utilities. The left side of the diagram shows how files are uploaded via HTTP; the right side uses FTP to upload files. There is a data-cache attached to each of the uploading programs.

The bottom portion of the previous figure performs the actual conversion of data once the data set is uploaded to GMS using either HTTP or FTP. The AML programs for the three conversion utilities are used to control the ARC/Info commands that convert DLG, TIGER/Line and DXF data to e00 format (Figures 4.17, 4.18 and 4.19, respectively). The AML programs are all fairly straight forward in that they take in command line arguments, convert the Arc/Info coverages to their respective data formats and return the files to the user for download (ESRI, 1994a; ESRI, 1994b).

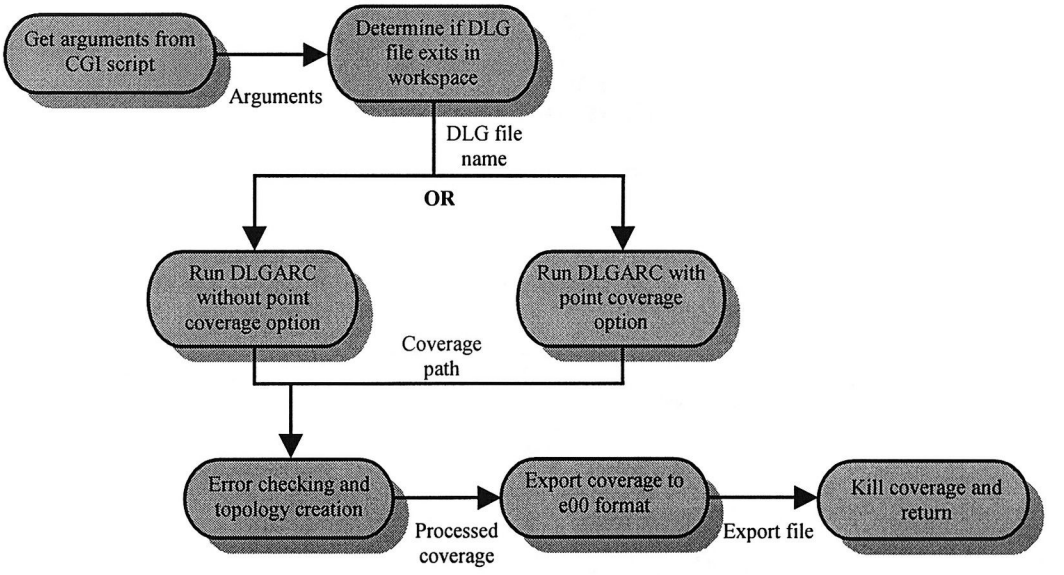


Figure 4.17 Flow diagram for the AML used in the DLGARC geospatial data-conversion utility. The conversion command provides the option to create a point coverage from the degenerate line record in the DLG data.

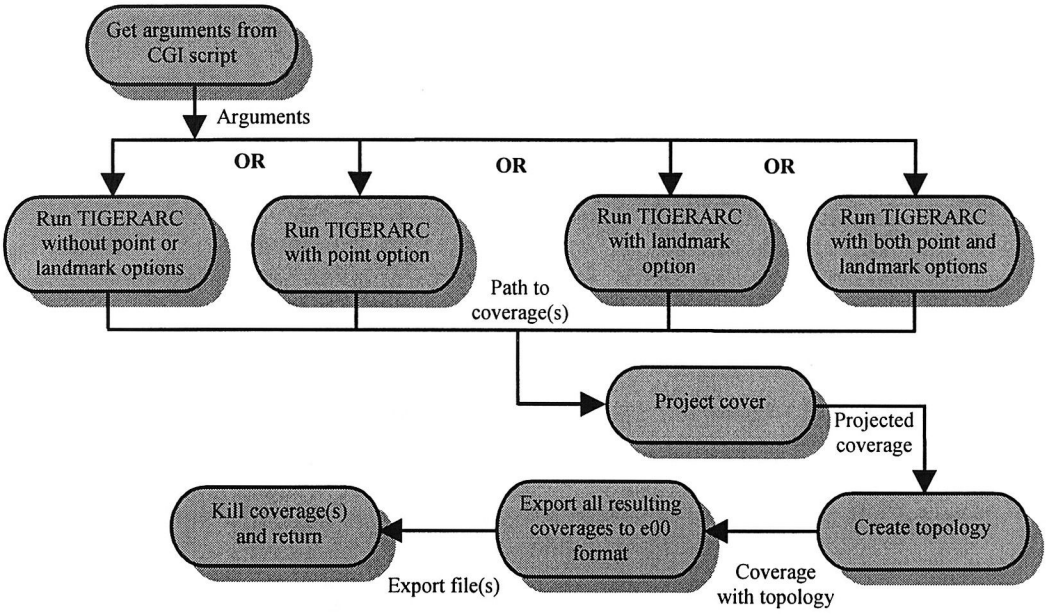


Figure 4.18 Flow diagram for the AML used in the TIGERARC geospatial data-conversion utility. The conversion command provides the option to create coverages containing either point features, landmark features or both. The utility performs also optional coverage projection and topology creation.

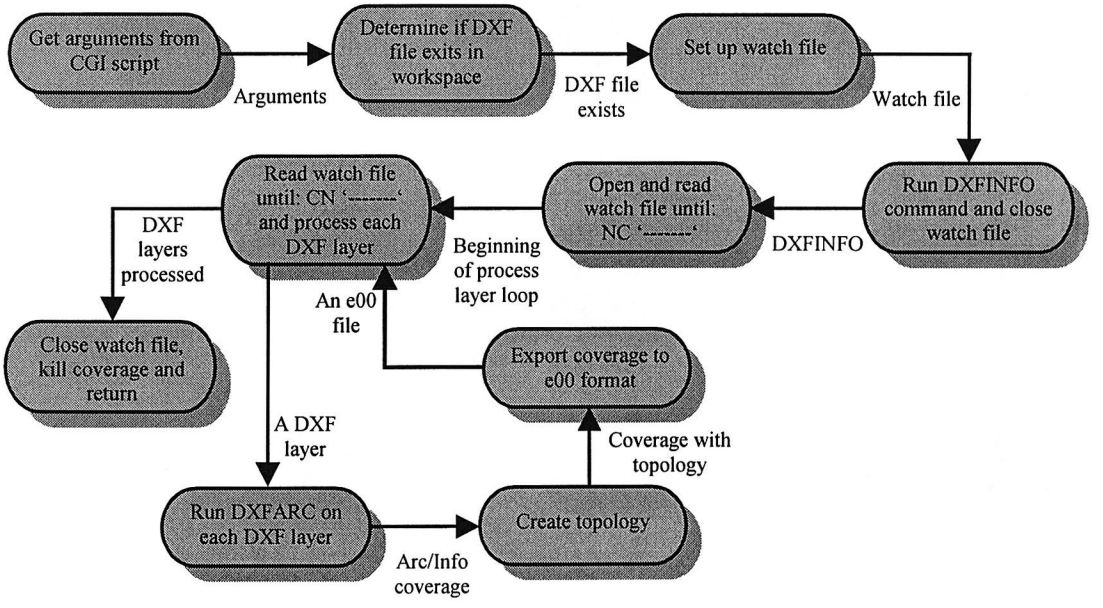


Figure 4.19 Flow diagram for the AML used in the DXFARC geospatial data-conversion utility. The conversion command must automatically set-up a 'watch' file that contains information from a DXFINFO command. The file is examined for DXF layers and an Arc/Info coverage is created for each layer.

The following conversion utilities are used to convert Arc/Info coverage in e00 format to DXF, DLG or TIGER/Line data formats (Figures 4.20, 4.21 and 4.22). They perform the opposite operation to the ones described in Figure 4.16 by uploading one or more e00 files from the Web and converting them to DXF, DLG or TIGER/line data formats. The converted files are then returned to the client for download. The conversion process outlined in Figures 4.20, 4.21 and 4.22 are somewhat more complicated then the three Web translators mentioned above. The ARCDXF, ARCDLG and ARCTIGER conversion utilities require anywhere from one to four e00 files to be uploaded individually by the script. It is important to note that these uploaded e00 files need not originate from the same server, but in fact, can be retrieved from different servers located in different locations on the Internet.

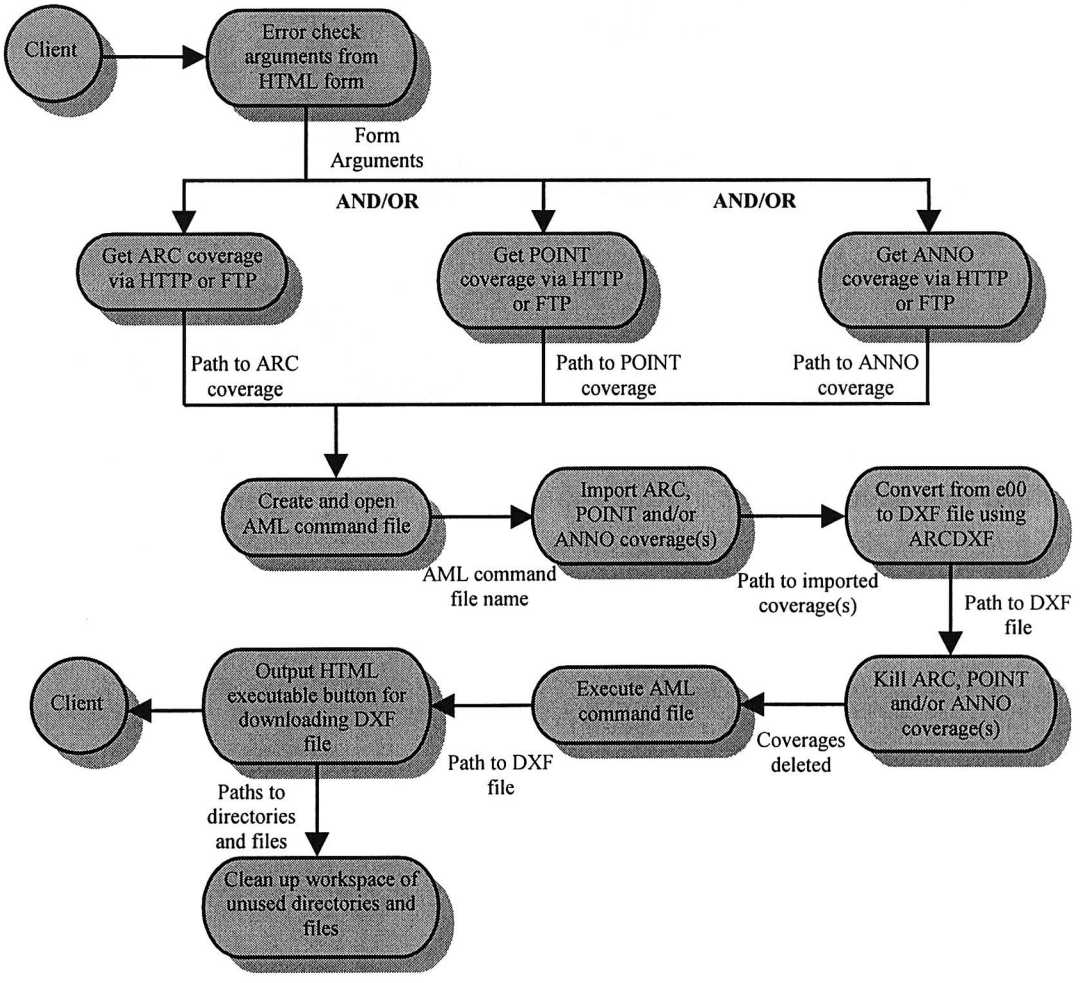


Figure 4.20 Flow diagram for the ARCDXF geospatial data-conversion utility. The utility uploads to GMS separately one or more arc, point and annotation coverages. They are used to create a DXF data set containing one or more DXF layers.

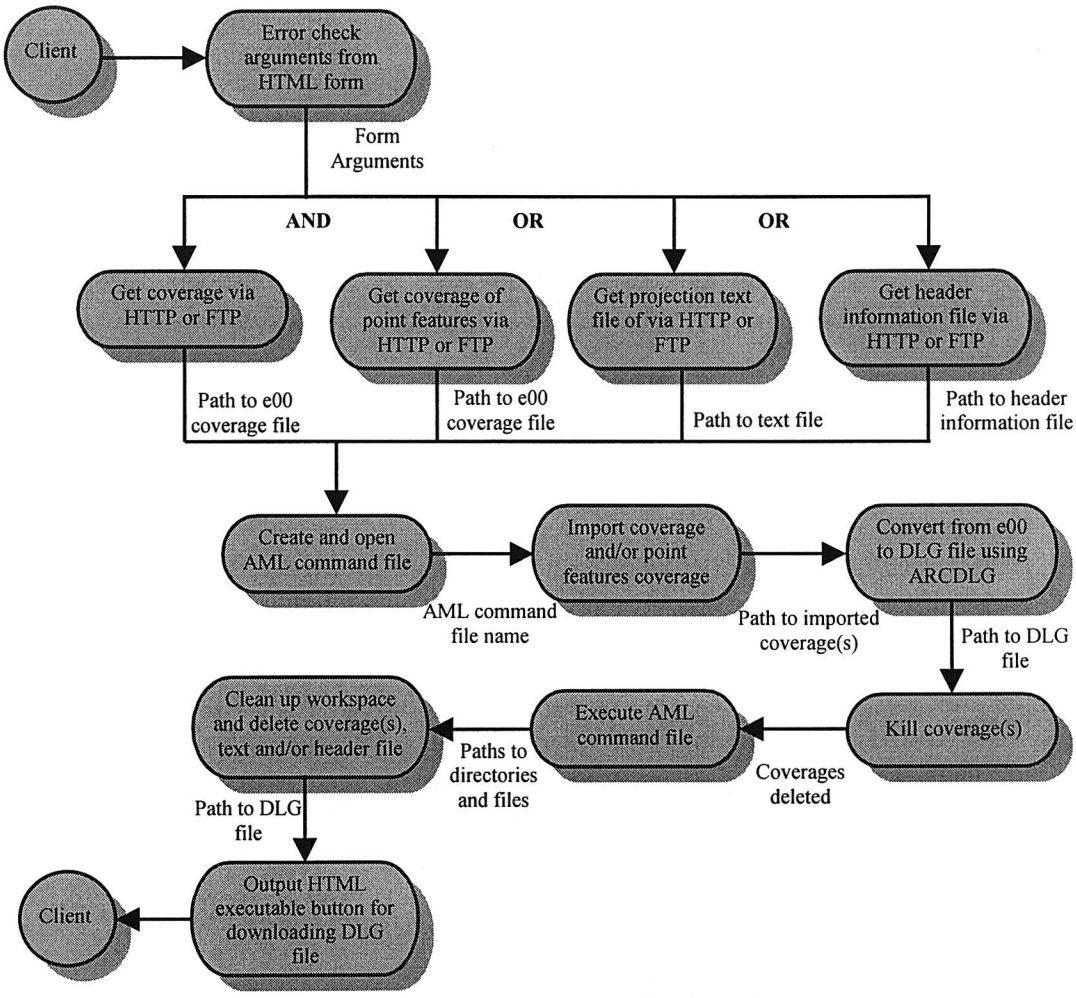


Figure 4.21 Flow diagram for the ARCDLG geospatial data-conversion utility. The utility uploads a coverage, an optional point coverage, a file containing projection parameters and a DLG header file.

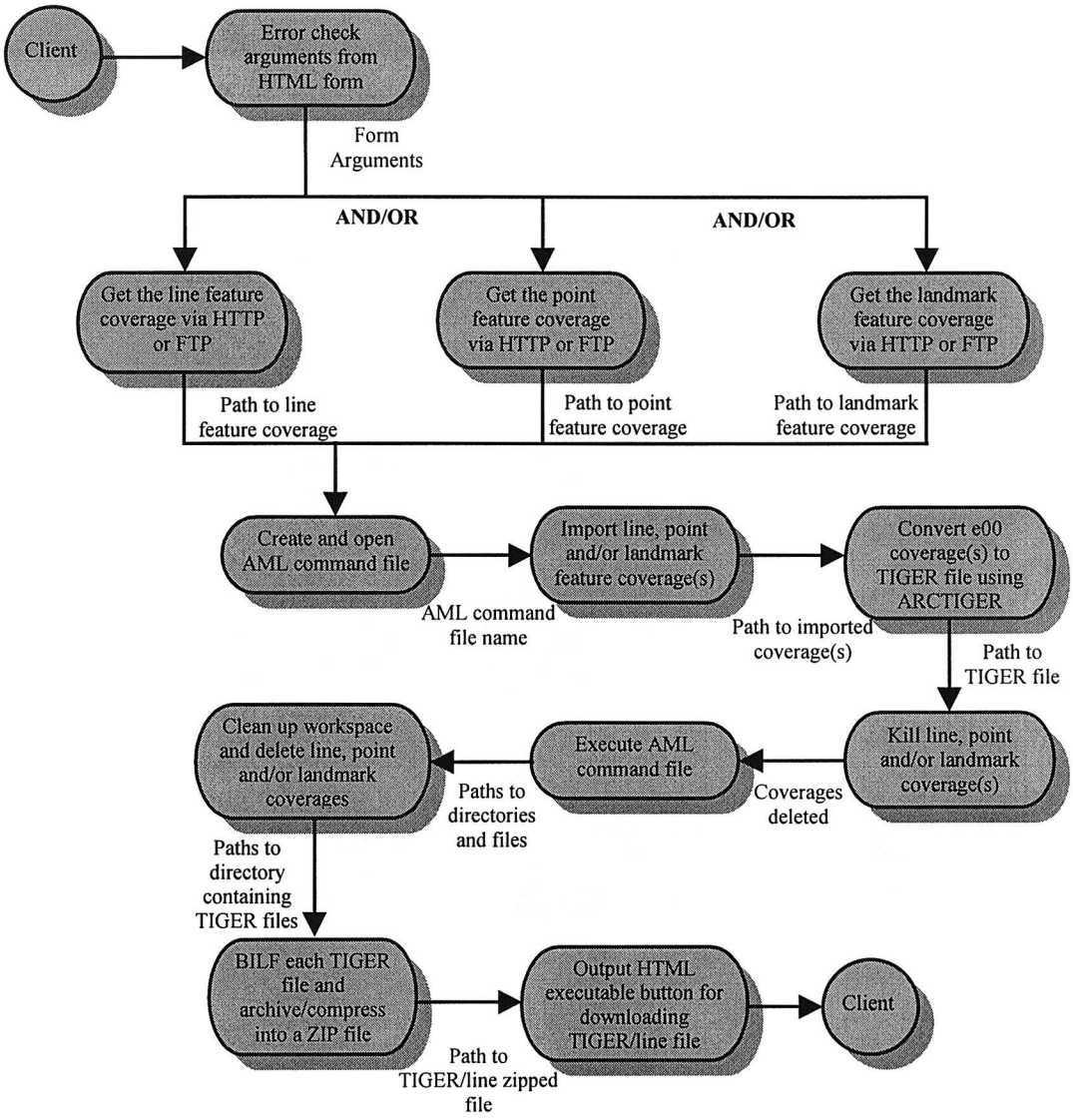


Figure 4.22 Flow diagram for the ARCTIGER geospatial data-conversion utility. The conversion command accepts a coverage containing line, point and/or landmark features. They are imported and subsequently converted to a TIGER/Line data sets.

The e00 file decompression and plot utilities demonstrate how some fundamental GIS operations are used over the Web. The first utility, shown in Figure 4.23, decompresses e00 files by first uploading a compressed e00 file, importing it to a coverage and re-exporting the coverage to a new uncompressed e00 file. Uncompressed e00 files are simply GIS data in ASCII format. This e00 file decompression utility allows users to convert Arc/Info GIS data into standard ASCII format for use in their own GIS software.

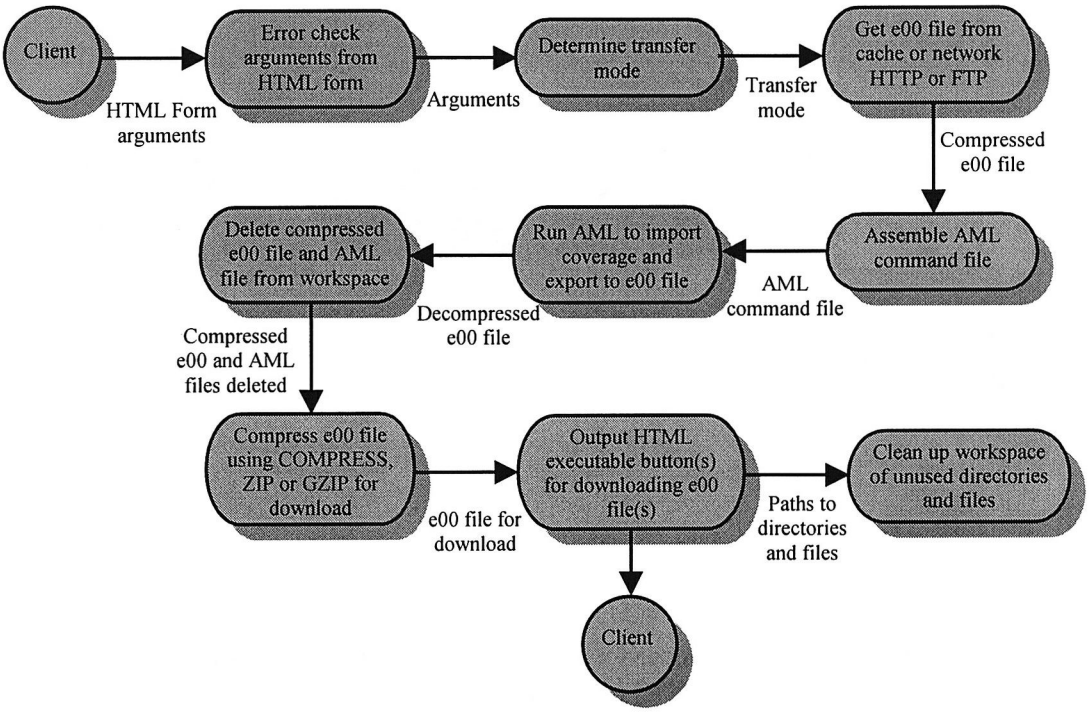


Figure 4.23 The e00 file decompression utility. The utility converts coverages to standard ASCII format over the Web for use in other types of GIS software.

Similarly, the ARCPLOT utility shown in Figure 4.24 allows Arc/Info GIS data to be uploaded as e00 files, imported to coverages, plotted to a variety of different graphic formats and displayed to the Web browser.

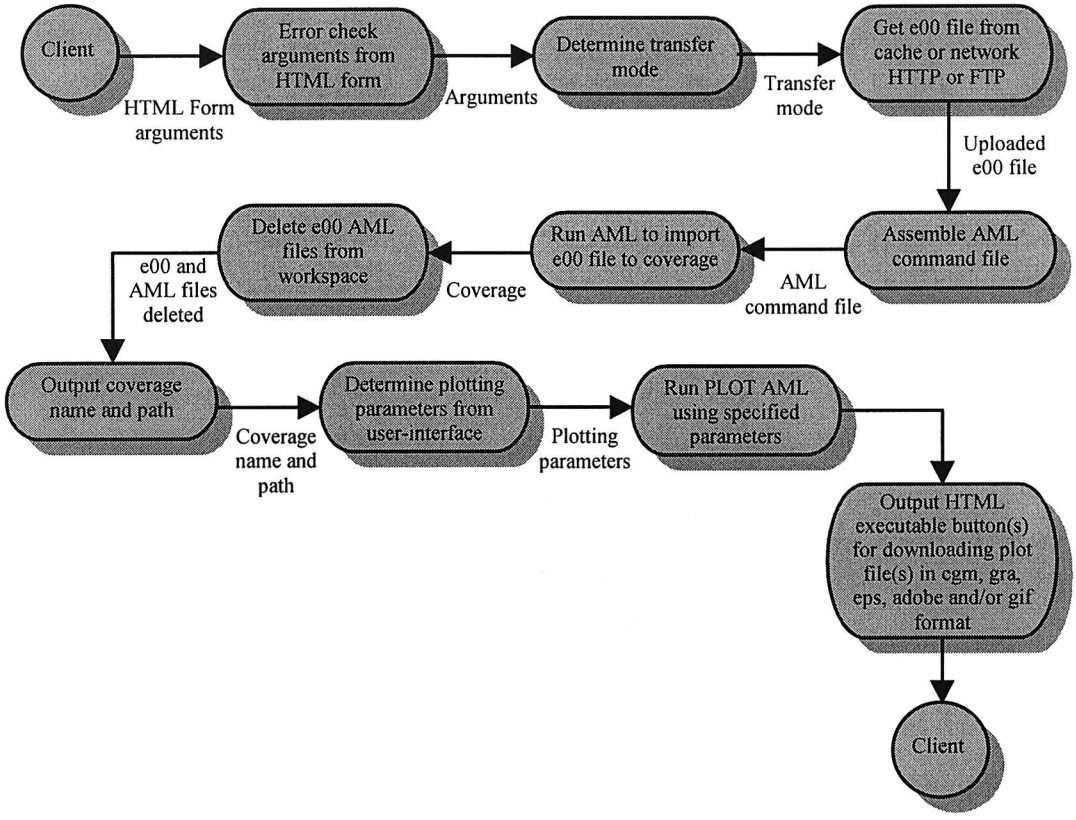


Figure 4.24 The Web based ARCPLOT utility. The utility uploads a coverage in e00 format, converts it to a CGM, GRA, EPS or ADOBE plot file and returns it to the client.

4.2.3 Software implementation

The following sub-sections provide a detailed description of the implementation of all six conversion commands as well as the on-line e00 file decompression and plotting utilities.

4.2.3.1 The user-interface

There are a number of important user-interface components that are used repeatedly for the utilities discussed in this section. One interface component, in particular, is the HTTP and FTP file upload program. Figure 4.25 shows the general layout of the interface used for this program. In this instance, the program is used as part of the interface for the e00 file decompression utility.

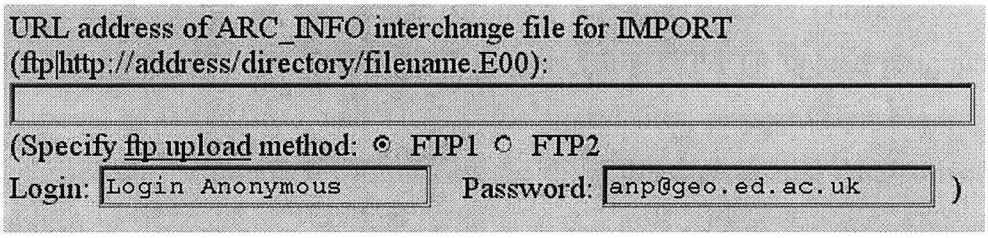


Figure 4.25 HTTP and FTP files upload facility used in all the on-line GMS utilities.

The program consists of three text fields and two radio buttons. The longer text field is used to enter the URL address that references an on-line geospatial data resource. It has a maximum length of 250 characters, which should be sufficient for most URL addresses. Below this field is additional information that is required by the CGI script to perform a standard 'get filename' FTP operation. The information supplied by the radio buttons represents two methods by which the GMS FTP client retrieves files from the directory trees of remote FTP sites. Both methods use the FTP 'get' operation. The first radio button retrieves the file with a single command that includes the filename and its path:

get dir1/dir2/...dirN/filename

The second radio button performs the same operation by using separate command lines for each directory path followed by a carriage return. This second methods was constructed to ensure compatibility with certain FTP server implementations.

cd dir1<RT>
cd dir2<RT>
.
.
.
cd dirN<RT>
get filename

Uploading FTP files also requires login and password information that is entered into the two fields below the radio buttons of Figure 4.25. The two fields allow users to access files from anonymous FTP sites as well as their own FTP sites protected by a username and password.

The geospatial data-conversion utilities outlined in Figure 4.16 have similar characteristics as far as the design of their user-interfaces is concerned. The DXFARC, DLGARC and TIGERARC possess the HTTP/FTP upload program for the retrieval of DXF, DLG, and TIGER/line zipped files, respectively. Figure 4.26 summarises all the interface components for these three geospatial data-conversion utilities.

Hypertext help information	Data set upload facility	Additional conversion arguments	Coverage processing	Export of coverage	Submit button

Figure 4.26 The interface components of the DXFARC, DLGARC and TIGERARC conversion utilities.

Each of the conversion utilities are accompanied by help information in the form of hypertext links to the actual Arc/Info help documents describing the conversion commands used in each utility. The user-interface for each conversion utility also allows the client to enter a specified set of optional or mandatory arguments required by the Arc/Info conversion commands. Optional coverage processing is also a feature of these three conversion utilities and, in each case, there are various options related to the export of the converted geospatial data from coverages to e00 files.[3]

The user-interfaces for the second set of conversion utilities, outlined in Figures 4.20 to 4.22 inclusive, are constructed rather differently from the three utilities just

[3] For an in-depth explanation all the conversion utilities mentioned in this section as well as the Arc/Info EXPORT command, the reader should consult the help documentation contained within the GMS Web site.

described. The ARCDXF, ARCDLG and ARCTIGER conversion utilities are able to upload one or more e00 and/or other files from the Internet and use them within the conversion process. For example, the ARCDXF conversion utility requires one or more of three different Arc/Info coverages (arc, point and/or annotation) that are uploaded as e00 files with the HTTP or FTP upload program. Therefore, the ARCDXF user-interface consists of three HTTP/FTP upload forms of the type shown in Figure 4.25, one for each type of coverage. The ARCDLG conversion utility is no different in this respect requiring four upload-forms that include:

- a mandatory coverage containing polygon, line or node features,
- an optional coverage containing point features,
- a text file containing projection parameters,
- a file containing header information for the resulting DLG file.

Similarly, the ARCTIGER conversion utility requires three HTTP/FTP upload forms. These include coverages containing the basic line features (and attribute data), point features (representing polygon labels) and landmark features.

The user-interfaces of the three conversion utilities described in the above paragraphs also contain some the components shown in Figure 4.26. These components include hypertext links to help information for the ARCDXF, ARCDLG and ARCTIGER Arc/Info conversion commands. The utilities also contain text fields and radio buttons for various conversion arguments, among which include the output DXF, DLG and TIGER/line file names.

The e00 file decompression and ARCPLOT utilities are rather similar in user-interface design to the conversion utilities already described in this section. The e00 utility possesses hypertext links to help documentation of the Arc/Info EXPORT and IMPORT commands as well as a HTTP/FTP upload program for the e00 file and some additional textual form fields for command-line arguments. There is also a series of radio buttons that allow the user to select an alternative type of file compression before returning the uncompressed e00 file to the user (Figure 4.27).

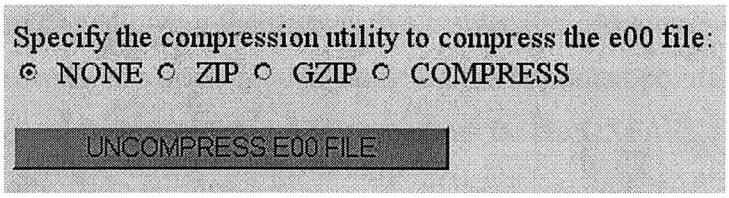


Figure 4.27 A portion of the user-interface for the e00 decompression utility that shows the different compression programs available to compress the resulting e00 ASCII file.

The ARCPLOT utility allows the user to specify the URL address of an on-line e00 file. Alternatively, another field is provided for path, directory and file information of a coverage that already exists on the GMS system. The remainder of the user-interface consists of checkboxes and text fields which allow the user to specify the coverage features for plotting, the line or marker for shadings and the format of the resulting plot file. See Figure 4.28 below:

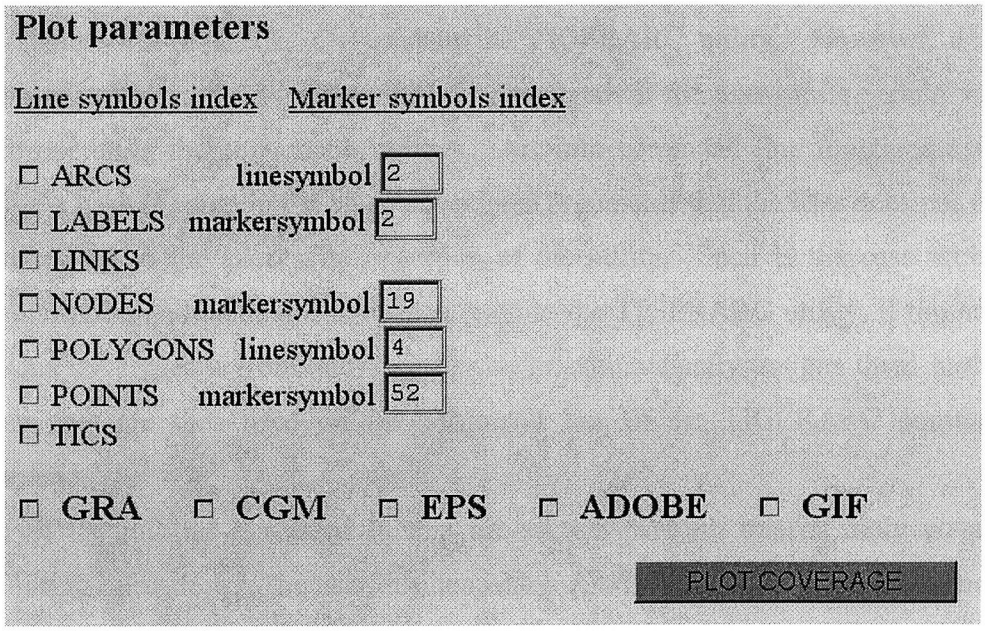


Figure 4.28 A portion of the user-interface for the ARCPLOT utility showing the plot parameters that include the feature type, symbol type and output file format.

4.2.3.2 The DXFARC, DLGARC and TIGERARC conversion utilities

The GMS components that convert DXF, DLG and TIGER\line to e00 files are designed using roughly the same code. Because of the similarities in user-interface

design, all three conversion utilities consists of CGI scripts that operate in basically the same manner by allowing their respective conversion commands to be operated from a Web browser. This sub-section describes the important points with regard to the CGI scripts and their associated AML programs for the DXFARC, DLGARC and TIGERARC conversion utilities.

The CGI scripts of each of the utilities perform a number of checks on the range and format of each HTML form field value upon receiving the arguments from the user-interface. In addition to determining whether or not a URL was entered, the DXFARC utility ensures that the width of the DXF text and attribute fields are within the proper range of 0 to 255 characters. The DLGARC utility ensures both a coverage name and URL are supplied by the user. The utility also supplies default values to fields that are left blank such as in the case of the optional point coverage that is sometimes used while converting data from DLG to e00 format. Even for numeric fields, default values are always supplied and checks are conducted to ensure the user does not supply character values where numeric ones are required. The same sorts of checks are used in the TIGERARC utility. However, default values are also required for the Arc/Info commands if the user omits certain values when operating the conversion utilities. Arc/Info command line functions usually require a hash (#) sign in the place of optional arguments that the user does not deem as necessary to the successful operation of the utility. Such is the case with the optional point and landmark coverages used in the TIGERARC utility; if the names of these coverages are omitted, then the script automatically assigns them hash (#) values that are then used on the command line of the TIGERARC conversion command.

All of the utilities discussed in this sub-section perform one or more coverage processing operations. The user decides which Arc/Info function to perform and the operation is conducted after each conversion utility transforms the DXF, DLG and TIGER/line data into Arc/Info coverages. The function of the CGI scripts is to interpret which operation the user selects and to supply this information to the AML programs. The AML programs process and convert the coverages. The DXFARC utility allows the user to perform three coverage-processing operations, which include:

- CLEAN/POLY to generate a coverage with correct polygon topology,
- BUILD/LINE/JOINITEM which creates or updates a line attribute table for a coverage and then merges two INFO data files based on a shared item if more than one INFO file is detected as a result of the DXF-to-coverage data-conversion.
- BUILD/POINT/JOINITEM, which performs exactly the same operation as build/line/joinitem with the exception that a point attribute-table, is created for the coverage.

The script determines operations by initially assigning a value of 0 to internal DCL[4] variables and then changing the value to 1 for the selected operation. The values of these internal variables are then forwarded to the DXFARC AML program as command line arguments that execute the Arc/Info commands responsible for coverage processing. The coverage processing operations for the DLGARC utility are the same as those used for the DXFARC utility and also operate in a similar manner.

The TIGERARC utility performs coverage processing operations somewhat differently than the two other utilities just mentioned. Coverages that are converted from TIGER/line data can have their co-ordinates altered to another projection. The TIGERARC interface provides a checkbox to initiate the Arc/Info PROJECT command and an HTML text-area field where the user specifies the parameters used for the projection operation. The projection parameters are placed in a text file and its file name is supplied to the conversion command. It is important that a projection file is supplied if the user intends to conduct such an operation. If no file is detected by the PROJECT command, Arc/Info defaults to an interactive session. Such interactive sessions can not operate over the Web because of the request/response model upon which it is based (transactions are atomic)[5].

[4] Digital Command Language. A substantial portion of the GMS conversion utilities consist of utilities developed in this language.

[5] This aspect of the Web was briefly discussed in chapter 2. One can also refer to Thomas (1996) for a more in-depth explanation of the technology associated with the Internet.

The user is able to specify a series of operations that build topology from the coverages converted from TIGER/line files using the TIGERARC utility. Some of these operations are outlined as follows:

- BUILD/LINE, create or update the line attribute table of a coverage;
- RENODE, renumbers nodes for coverage arcs and updates values for FNODE# and TNODE# columns in the AAT table when they exist;
- ADDITEM, inserts blanks or zeros to item in an INFO data file.

The above operations use the Arc/Info database management system (INFO) to calculate and subsequently modify the coverage's AAT table. The coverage is then cleaned; labels are created and built once again before being exported to e00 format and returned to the user. The entire process consists of a standard procedure (Environmental Systems Research Institute, 1991) for converting TIGER/line data files to Arc/Info coverages. It demonstrates how relatively complicated GIS operations can be performed over the Web while continually delivering feedback to the Web browser.

All of the conversion utilities are comprised of separate AML programs that are executed from within each of their respective CGI scripts by constructing and executing an AML command file. The file is first created and opened to receive DCL 'write' statements:

```
$ open/write tempfile 'aml_command'
```

Statements to enter the Arc/Info shell, run the AML program supplying the necessary arguments, exit the shell and close the command file are as follows:

```
$ open/write tempfile 'aml_command'
$ write tempfile "$ ARC"
$ write tempfile "&RUN anp_disk:[anp.scripts]com.aml ",argv1," ",argv2," ",argv3,"...
$ write tempfile "quit"
$ close tempfile
```

The command file is then run from within the script:

```
$ @'aml_command'
```

As the AML program runs, each of the commands and the resulting output is echoed back to the browser. It was a major implementation decision to allow the user to inspect the execution and output of the AML programs. Due to the excessive length of time required to perform data-conversions, the user needs some sort of feedback in order to monitor the progress of the operation. Otherwise, the user may mistake a blank screen as a network problem and subsequently abort the entire process. There are easier ways of executing programs from within CGI scripts, but the utility of this method becomes more apparent in the next section when the ARCDXF, ARCDLG and ARCTIGER conversion utilities are discussed.

Each of the AML programs consists of relatively small pieces of code containing successive Arc/Info shell commands with their arguments. The programs can be generally summarised in the following steps:

- Assign command-line arguments from the script to AML variables.
- Determine if the uploaded data set exists in the Arc/Info workspace.
- Convert the data set to coverage(s) using any options available.
- Check for any coverage errors by confirming the existence of mandatory tables.
- Create topology for the coverage(s).
- Export the coverage(s) to e00 format.
- Kill the coverage(s) to conserve disk space and return to the CGI script.

Figures 4.17, 4.18 and 4.19 show the above steps in more detail for each of the conversion utilities discussed in this sub-section.

A major complication in the design of these AML programs is the complexity inherent in the conversion of DLG, DXF and TIGER/line data to Arc/Info coverages. The conversion process for each of these utilities is by no means straightforward. For each Arc/Info conversion command there are a number of input options which needed to be addressed in order to provide a substantial degree of flexibility for the user and to ensure the utilities function at an optimal level. For example, the DLGARC utility allows the creation of a single optional point coverage from degenerate Line (L) records in the DLG data set (Figure 4.29).

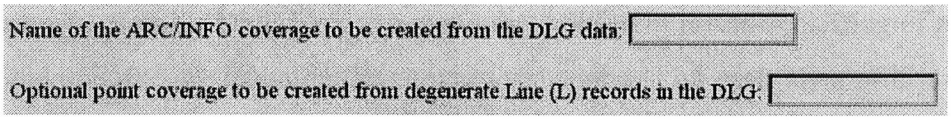


Figure 4.29 A portion of the DLGARC user-interface showing two fields that require the names of the converted coverage and the optional point coverage.

A '#' value is assigned to the associated variable in the CGI script and AML program if the field requiring the name of the optional point coverage is empty. If a coverage name is specified for the optional point coverage field, the file name is used in the DLGARC conversion command, which is executed accordingly as shown in Figure 4.17.

The TIGERARC conversion command (Figure 4.18) requires the name of the output coverage which contains the basic line features and attribute data from the set of TIGER/Line files. The command also requires the names of two optional coverage names, one to contain point features that represent polygon labels and the other to contain landmark features. Figure 4.30 shows a section of the TIGERARC user-interface where these three coverage names are entered.

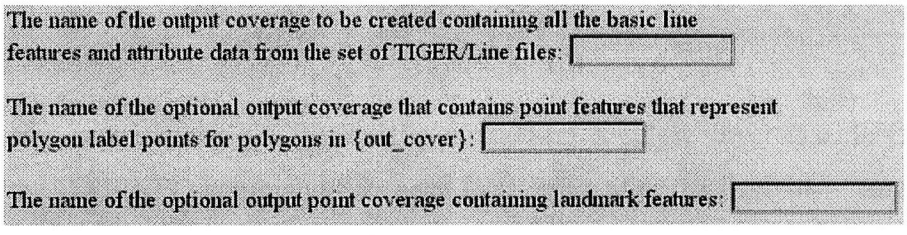


Figure 4.30 A portion of the TIGERARC user-interface showing the optional coverage fields for the point and landmark features.

The CGI script determines which optional coverage names are selected and substitutes a '#' value for those fields left blank. The arguments are supplied to the TIGERARC AML program of Figure 4.18 which performs the TIGERARC command converting the TIGER/Line files to one or more e00 files.

As mentioned earlier, the DXFARC conversion command operates somewhat differently than the two conversion commands described above. The DXFARC AML program was patterned from the AUTODXF.AML program Greenberg (1994) after considerable changes for use over the Web. The DXFARC Arc/Info command is designed to operate interactively with the user because a DXF data file is

composed of different layers, which must be individually converted to coverages. For example, a single DXF file can contain three different layers consisting of points, lines and annotation. Running the conversion command on each layer will produce coverages of points, lines and annotations, respectively. In order to determine the layers that exist within a particular DXF file, the Arc/Info command DXFINFO is run on the DXF file printing a list of the layers contained within the data set to the screen. The user must manually type the names of these layers when prompted by the DXFARC conversion command.

Both the DXFARC and DXFINFO command were designed for interactive use within the Arc/Info shell. The challenge was to design an AML program that executed both commands automatically. Figure 4.19 outlines the necessary steps used to convert a DXF file to one or more e00 files without user interaction.

Prior to running the DXFINFO command, a watch file is set up which directs all screen output to a specified file. Therefore, the output of the DXFINFO command is written to a temporary file and stored for use later in the program.

LAYER NAME	ARCS	POINTS	TEXT	ATTRIB	INSERT	TEXT LEN	DEF COLOR	DEFAULT LINETYPE
0POINTS	0	23	0	0	0	0	0	
ALL LAYERS	0	23	0	0	0	0		

The output represents a DXF data file containing a single layer (i.e. 0POINTS). Notice the DXFINFO command lists each DXF layer after the first '-----' and before second the '-----' lines. The DXFARC command can be run automatically on each DXF layer by simply reading each layer name from the watch file between the two bounding lines just described. So the AML program would first read the watch file until the first '-----' is reached and then proceed to read the names of each DXF layer. A subroutine is initiated upon reading each layer name, which creates the coverages, builds the necessary topology and exports the coverages to e00 format. The process is repeated until the second '-----' is reached thereby ending the AML program and returning control to the script.

The final portion of the CGI script of each conversion utility essentially cleans up the directory space and provides an HTML executable button to download one or more of the converted e00 files. Both the uploaded data file and the AML command

file are deleted from the workspace along with any remaining files or coverage directories created during the data-conversion process. The HTML executable buttons used in all the data-conversion utilities are created by their respective CGI scripts and subsequently sent to the browser. HTML buttons created in this manner can be summarised by the following diagram:

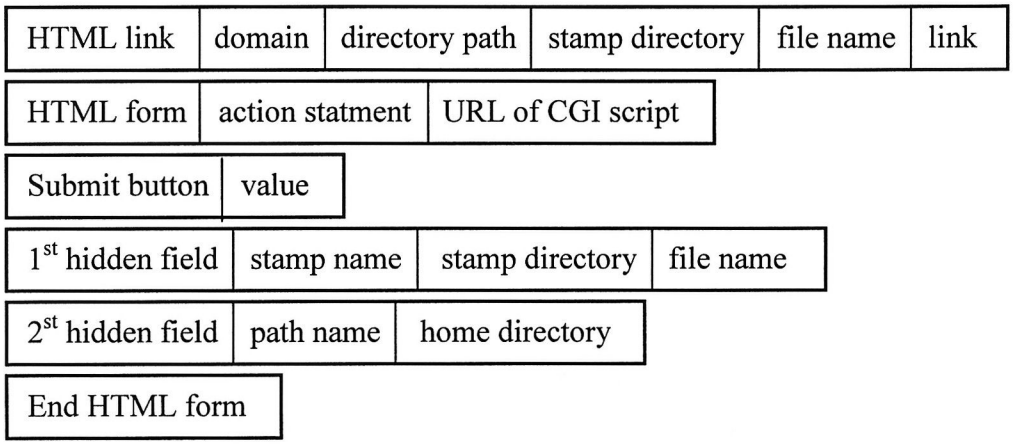


Figure 4.31 The structure of an HTML executable button. The top series of boxes is a hypertext link to the data sets. The executable button is a simple HTML form with only two types of form elements, a submit-button and various hidden fields. The hidden fields contain command-line arguments, which are sent to the application referenced in the second line of the above figure.

The top set of boxes in the above figure shows a hypertext link that accompanies the executable button, which can be used to re-enter the converted data file into some of the other GMS utilities. The link allows the user to access newly converted geospatial data sets from the GMS server and convert them once again to another data format. The data sets can also be sent to other on-line applications that reside either on GMS or other remote severs. The ability to inter-convert between geospatial data formats using a common data format like e00 or SDTS file is a fundamental requirement of the data-switchyard. The concept as it applied to this research is shown in Figure 4.32 below:

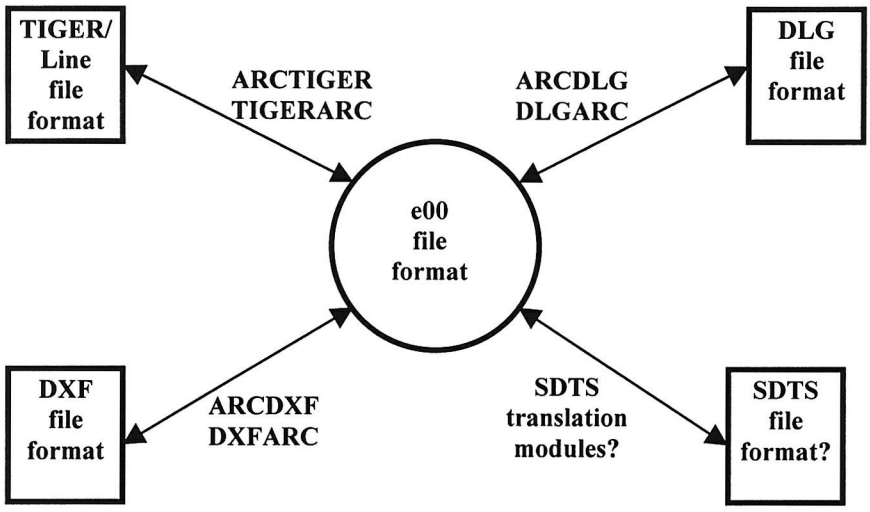


Figure 4.32 Data-switchyard architecture (from Pradhan and Gittings, 1996).

Figure 4.32 shows that an e00 file format can be used as a common data format to inter-convert between TIGER/Line, DLG and DXF file format. If the appropriate conversion utilities exist, an SDTS file can also be used as common data format when converting from one file format to another.

The remaining boxes of Figure 4.31 show the construction of a typical HTML executable button created by the CGI script. The button consists of a standard HTML form without any input fields except for a single 'submit' button'. The 'submit' button takes the information within the two hidden fields and sends it to another CGI script (specified by the URL address) whose sole purpose is simply to download the remote data set to the user's system in either ASCII or binary formats. Note that the DXFARC conversion utility would at times require more than one executable button written to the browser because a single DXF file can result in one or more e00 files following conversion.

4.2.3.3 The ARCDXF, ARCDLG, ARCTIGER conversion utilities

The conversion utilities discussed in this section are designed to convert Arc/Info e00 files to either DXF, DLG or TIGER/line geospatial data formats. The utilities are somewhat more complicated to design than the ones described in the previous sub-sections because of their ability to upload one or more e00 files and incorporate

them within the data-conversion process. All the AML code for each of these conversion utilities is generated on-the-fly by their respective CGI scripts.

Following the brief description (in section 4.2.3.1) of the user-interfaces of the ARCDXF, ARCDLG and ARCTIGER conversion utilities, this sub-section elaborates on what occurs as the values from the form fields are received by the CGI script. After error-checking the arguments, a major portion for each CGI script is devoted to uploading all the relevant e00 files specified by the user for inclusion in the data-conversion process.

Figures 4.20, 4.21 and 4.22 (and associated HTML interfaces) indicate the sort of coverages that are uploaded by each conversion utility. For example, the ARCDXF (Figure 4.20) utility initially requires one or more coverages of arcs, points or annotations. The mechanism by which the script determines which coverage is uploaded is shown in the sample DCL code below:

```
$ if FORM_FLD_ARCURL .EQS. "" then
$  FORM_FLD_ARCFILE = "#"
$  SCRATCH1 = "#"
$  GOTO FLAG1
$ endif
.
(upload arc coverage)
.
$ FLAG1:
$ if FORM_FLD_POINTURL .EQS. "" then
$  FORM_FLD_POINTFILE = "#"
$  SCRATCH2 = "#"
$  GOTO FLAG2
$ endif
.
(upload point coverage)
.
$ FLAG2:
$ if FORM_FLD_ANNOURL .EQS. "" then
$  FORM_FLD_ANNOFILE = "#"
$  SCRATCH3 = "#"
$  GOTO AML
$ endif
.
(upload annotation coverage)
.
$ AML:
```

Since there is a maximum of three e00 files to upload, this portion of the script is divided into three sections separated by flags (FLAG1, FLAG2, and AML). The IF-THEN block of statements that precedes each section tests whether or not a URL

address is provided to upload the coverage of that particular section. If, for example, the user had not provided the URL address to an arc coverage, then the first IF-THEN statement is 'true', the DCL symbol FORM_FLD_ARCFILE is given the default value of # and the script is advanced to the next section marked by FLAG1. Otherwise, a 'false' value for the IF-THEN statement uploads the arc e00 file via HTTP or FTP before proceeding onto the next section. In this manner, a minimum of one and a maximum of three coverages can be uploaded with the ARCDXF utility.

The same procedure to upload e00 files is used for the ARCDLG and ARCTIGER conversion utilities. The ARCDLG utility (Figure 4.21) initially requires the uploading of a single coverage in e00 format that can contain polygons, lines and/or nodes. In addition, three other file types can be uploaded and used within this conversion utility. These optional files include:

- A point feature coverage;
- A text file containing input projection parameters;
- Another text file containing information for the header of the DLG file.

The ARCTIGER utility (Figure 4.22) comprises of three sections capable of uploading three different e00 files. This situation is similar to the ARCDXF utility except that the initial line-coverage containing all the basic line features and attribute data is required for the successful creation of a TIGER/line file. The other two coverages associated with the ARCTIGER utility are optional. The first is a point-coverage containing the point features that represent polygon labels for the line/attribute coverage. The second coverage contains the landmark point features.

Once all the necessary e00 files have been uploaded from their particular Internet sites, the CGI scripts assemble the AML programs that convert the coverage(s) to DXF, DLG or TIGER/Line data format. Figure 4.33 is a generalised representation of the process by which the scripts:

- IMPORT the uploaded e00 files to coverages;
- perform the conversion;
- KILL the coverages used in the conversion process.

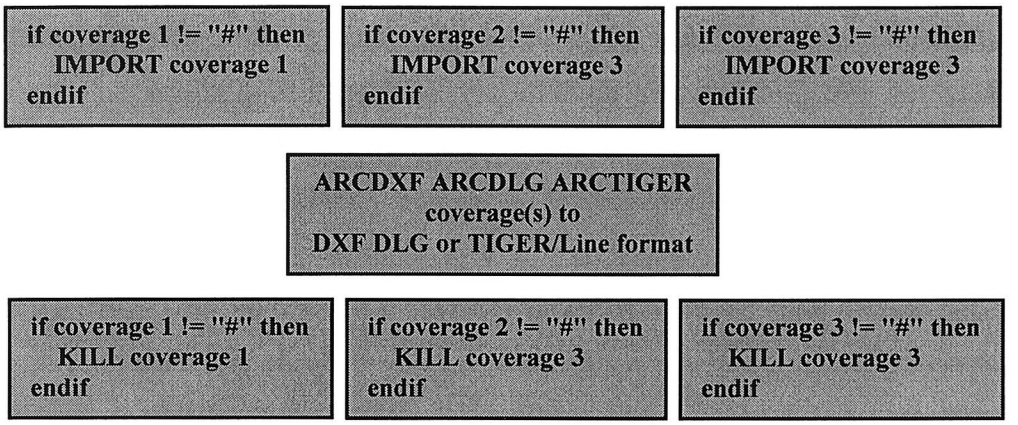


Figure 4.33 A set of boxes representing the AML commands used in the ARCDXF, ARCDLG and ARCTIGER conversion utilities. The top set of boxes determines the coverages to be imported and the middle box represents the data conversion. The bottom set of boxes determines the coverages to be deleted.

The script detects the existence of uploaded coverages by examining the value of their DCL variables. If the value of these variables is '#', then the coverage does not exist on the system and can not be imported or deleted using the Arc/Info IMPORT or KILL commands[6]. Otherwise, the AML commands shown in Figure 4.33 are written to a command file and executed from within the CGI script. The script then provides the resulting DXF, DLG or TIGER/Line files with executable buttons of the type shown in Figure 4.31 to allow users to download the files to their local servers.

4.2.3.4 The e00 file decompression and plotting utilities

Many of the features seen in the six conversion utilities described in the previous sub-sections are incorporated into a Web-based utility that transforms compressed e00 files to standard ASCII files. The e00 file decompression (Figure 4.23) utility uploads compressed e00 files and converts them to coverages using the IMPORT command. Then the coverages are converted back to e00 files using the Arc/Info EXPORT[7] command. The interesting feature in this utility is that the compression option for the EXPORT command is set to 'none' allowing users over the Web to

[6] IMPORT and KILL are Arc/Info commands. The IMPORT command converts an Arc/Info export interchange file (e00) to directory of files (a coverage) that can be read by the Arc/Info GIS. The KILL command deletes all data files and directories from the user's workspace.

[7] EXPORT converts a coverage, file or other supported data set to an interchange file for transfer to another platform running ARC/INFO.

convert their compressed e00 files to standard ASCII format. A utility such as the e00 decompression utility is extraordinarily useful for those people who do not have access to the Arc/Info GIS software. In addition to converting files to standard ASCII format, this utility can re-compress the files using standard compression algorithms (ZIP, GZIP or UNIX compress) to facilitate the transfer of large data files over the Web.

The ARCPLOT utility uploads a coverage in e00 format and plots one or more of its features to a number of different file formats which can then be either downloaded or viewed on the browser. A single coverage can possess a large quantity of information consisting of data that represents many different geometric features and attributes. Plotting is a process by which one or more maps can be produced from the tables of data that exist within a data set. Often, several different maps must be plotted to enhance the clarity between the different features and attributes. The resulting layers of data are then overlaid on one another to contrast and compare different phenomenon. This is the essence of plotting; it is a rather complicated GIS process using a single system. Plotting one's data sets over the Internet is just as difficult, if not more so, because the entire process must be conducted over a network of servers.

The GMS ARCPLOT utility incorporates a number of features to plot data sets using a standard HTML form user-interface (see Figure 4.28). Often, several plots must be constructed from a single data set making the process impractical to perform over the Web since the data must be uploaded each time a new plot is constructed. The ARCPLOT utility minimises this problem by uploading the entire data set once and supplying directory and path information so that the user can repeatedly access their data residing on the GMS server. Figure 4.34 below shows a portion of the user-interface that deals with this particular feature designed to save a considerable amount of time and bandwidth for plotting coverages over the Web.

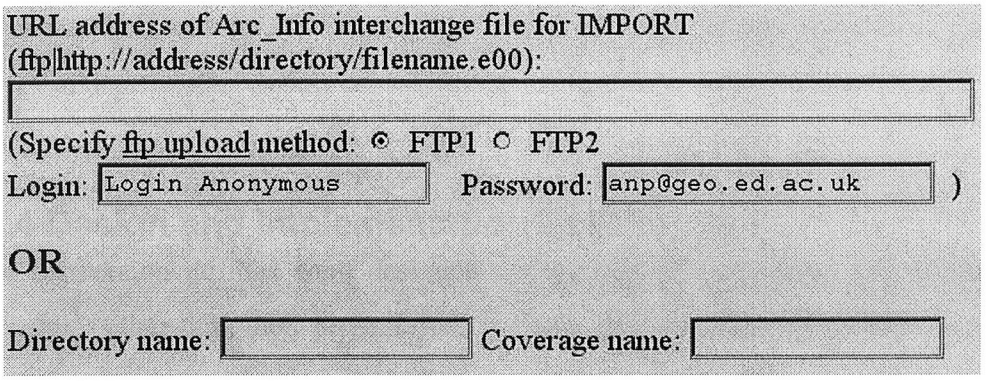


Figure 4.34 The upload facility for the ARCPLOT utility designed to upload the data set once and access it contents on the local GMS server thereafter.

The CGI script initially processes the plotting parameters by a series of IF-THEN statements that interpret user-commands and assigns appropriate values to the DCL variables for each parameter. For instance, each of the feature types (arcs, labels, links, nodes, polygons, points and tics) have their own DCL variables, which are initially given a value of zero. Upon selection by the user, the variables of the selected feature types are then given a value of one. This value is sent to the ARCPLOT AML program via the command line. The format of the plot file is selected in a similar fashion; the ARCPLOT utility can produce plots in GRA, CGM, EPS and ADOBE formats. The line and marker symbols are given numerical values by default that are then passed to the AML program by its command line interface. These numerical values can be altered if the user wishes to change the line or marker symbols for the plot.

The AML program performs a series of IF-THEN statements that performs each plotting function depending on the plotting parameters chosen from the user-interface. Using the ARCPLOT display command, the utility can set the proper format of the plot file as well as assigning a suitable name and file extension. The appropriate line or marker symbols are set and the map extents are defined before plotting each of the map features. A single run of the utility can produce the same plot file in four different formats if desired by the user. The important thing to note with this utility is that no additional proprietary software is needed except the Arc/Info GIS package. The ARCPLOT utility, in effect, turns a well-known GIS

operation into a process that can be performed over the Web using nothing more than a common Web browser.

4.2.4 Design and implementation constraints

The conversion utilities were designed without the user requiring any additional technology other than their present Web browser and the data they wish to convert. Though only six conversion programs (in pairs of three) were described in this section, the concept could be expanded to incorporate many other conversion algorithms. A Web site that can convert geospatial data between many different formats is of considerable benefit to geospatial data-users. Most of these users would have a limited amount of time and resources to purchase and learn anymore than a single GIS software package. Downloading and installing third party software packages has its own expense that must be considered in an organisation's overall strategy. From the user's perspective, there are considerable advantages of using Web-based conversion software over stand-alone software packages:

- low software maintenance costs;
- a short learning curve;
- use of standardised and easy-to-acquire technology;
- and the concentration of centralised and accessible services.

Keeping these conditions in mind, the development goal for creating the conversion utilities was to provide relatively complicated conversion services over the Web that provides a genuine service to the geospatial data-user community.

4.3 A geospatial Web robot

This section focuses on the design and implementation of a Web robot. The first step in designing a 'softbot' is to identify possible useful and feasible tasks for the software agent. Kautz et al, (1994) called this a bottom-up design process. The Web robot developed for GMS pro-actively searches for geospatial data sets by traversing HTML documents of a specified Web site, examining the file extensions of each hypertext link and storing the information in a text file and database to be viewed and/or queried later by the user. The utility known as Geobot is accessible from the local server and can be controlled on-line from a user's Web browser.

The robot is not directly linked to the metadata database described in section 3.1 (see Figure 1.1). In fact, it is provided with its own database, which stores the information collected by the robot as it traverses the hypertext links of a specified site domain. This information can later be queried much like other Web robots using Geobot's own search engine. However, Figure 1.1 shows that Geobot is linked to the metadata database indirectly via the GMS search engine described in section 4.1. More specifically, as metadata records are queried from the database using the GMS search engine, the search engine examines the *Network Address* attribute (appendix C) of the *Digital_Transfer_Information* table. If the value of the attribute (a URL address) links to a particular geospatial data file, then the search engine assigns any available conversion utilities (see section 4.2) to convert this data file to one or more other data formats (see the DXFARC HTML form button in Figure 4.11). If the value of the attribute links to an HTML document, then the search engine assigns the value to another HTML form button, which activates Geobot on the site domain containing the HTML document (see Figure 4.14). There are instances where a single metadata record is used to represent an entire data set consisting of multiple geospatial data files. Geobot demonstrates a mechanism to use the information contained within such metadata records in order to search and index entire site domains containing geospatial data sets.

4.3.1 Software design

Geobot has a fairly simple user-interface consisting of just four HTML pages. The pages and their hyperlinks are shown in Figure 4.35.

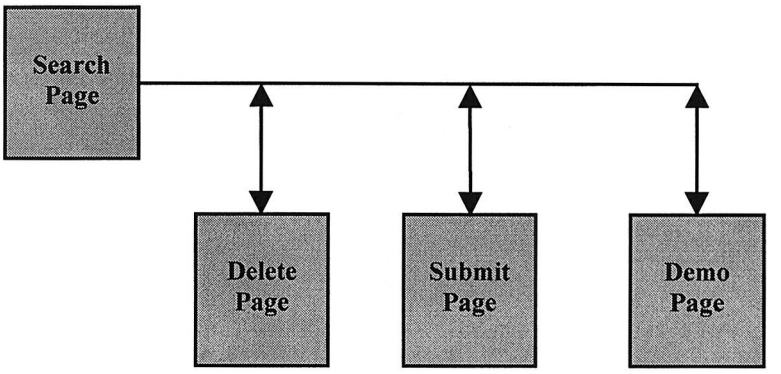


Figure 4.35 Geobot interface architecture.

The search page that is used to query the robot's database is linked to the three other pages. Each of these pages (delete, submit and demo) is linked back to the search page thereby providing the use with database query, database maintenance, geospatial data site search and demonstration capabilities.

The following programs are developed as support-utilities for the Web robot. Geobot is capable of amassing a large quantity of information, which is subsequently stored in its database. The database query utility allows clients to access this information on-line through a keyword search against certain database columns within the robot's database. As a larger number of users run the robot to traverse their Web sites, over time, a substantial amount of information accumulated in the database may decrease in relevance. Because of the excessive volatility of information on the Web, URLs referenced within Geobot's database records can become redundant. The database maintenance utility attempts to address this problem by giving the client the ability to delete records from the database that were originally placed there by the client. Flow diagrams where constructed for each utility as shown in Figure 4.36 and 4.37, respectively.

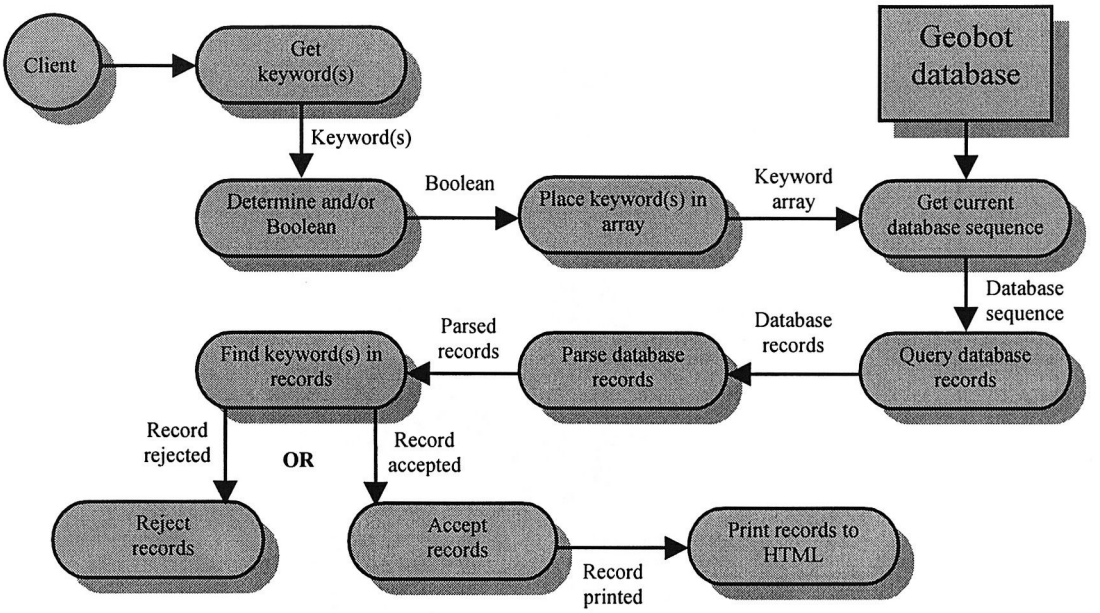


Figure 4.36 Flow chart of database query algorithm. This small search-engine is capable of boolean 'and/or' type queries.

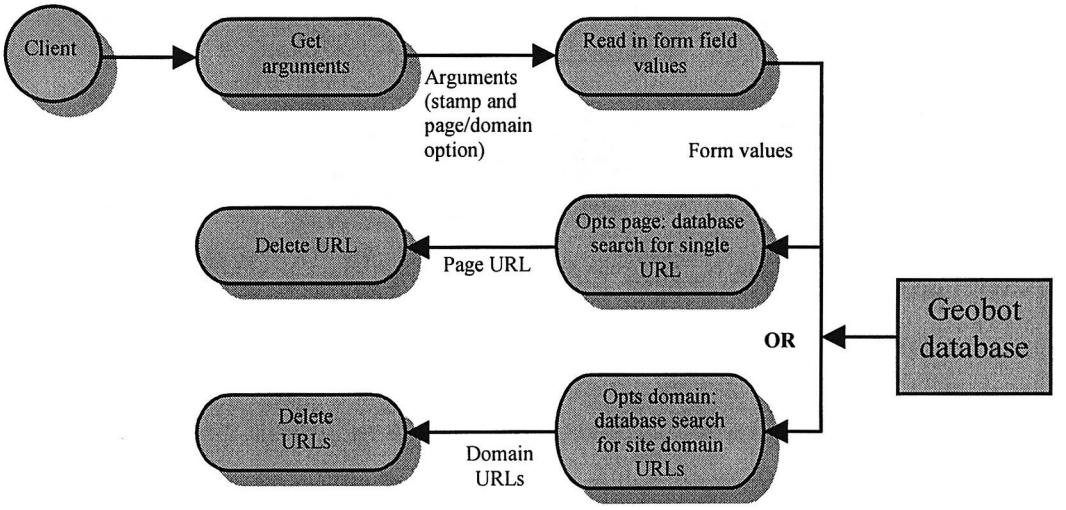


Figure 4.37 Flow chart of URL deleting algorithm capable of removing a single URL from the data from the database or an entire set of URLs that share the same domain (i.e. www.geo.ed.ac.uk) name[8].

The flow diagram for the robot crawling algorithm, shown below in Figure 4.38, provides Geobot with the ability to move from HTML page to HTML page while traversing a site domain. The mechanism revolves around a C language utility called

[8] See Mockapetris 1987a and Mockapetris 1987b for information on the naming scheme domain.

Fetch_HTTP (Jones, 1995) that connects to an HTTP server and performs a fetch of a URL. The URL is supplied to the program as a command-line argument.

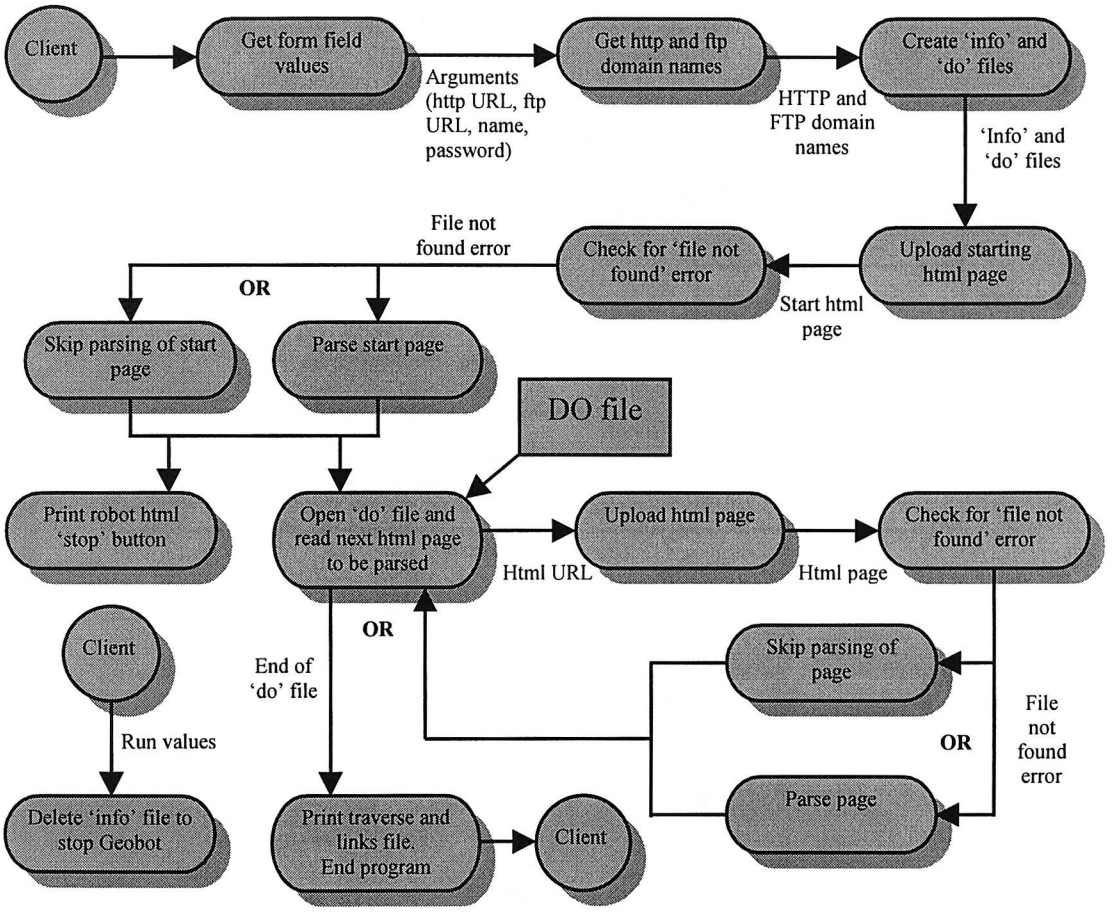


Figure 4.38 Flow chart of Geobot's 'crawl' algorithm that allows the Web robot to move from one HTML page to another.

Geobot's parsing algorithm directly complements the crawling algorithm. Each HTML page that is uploaded by the crawling algorithm is subsequently parsed of its URL addresses. The URL addresses are then categorised by file type. Those URLs ending with file extensions that match recognised geospatial data sets are placed in the database while those URLs that reference an HTML page are placed in a 'DO' file that the robot uses to traverse the site. URLs that are outside of the site domain boundaries or represent protocols other that HTTP are discarded. The process is shown in more detail in Figure 4.39.

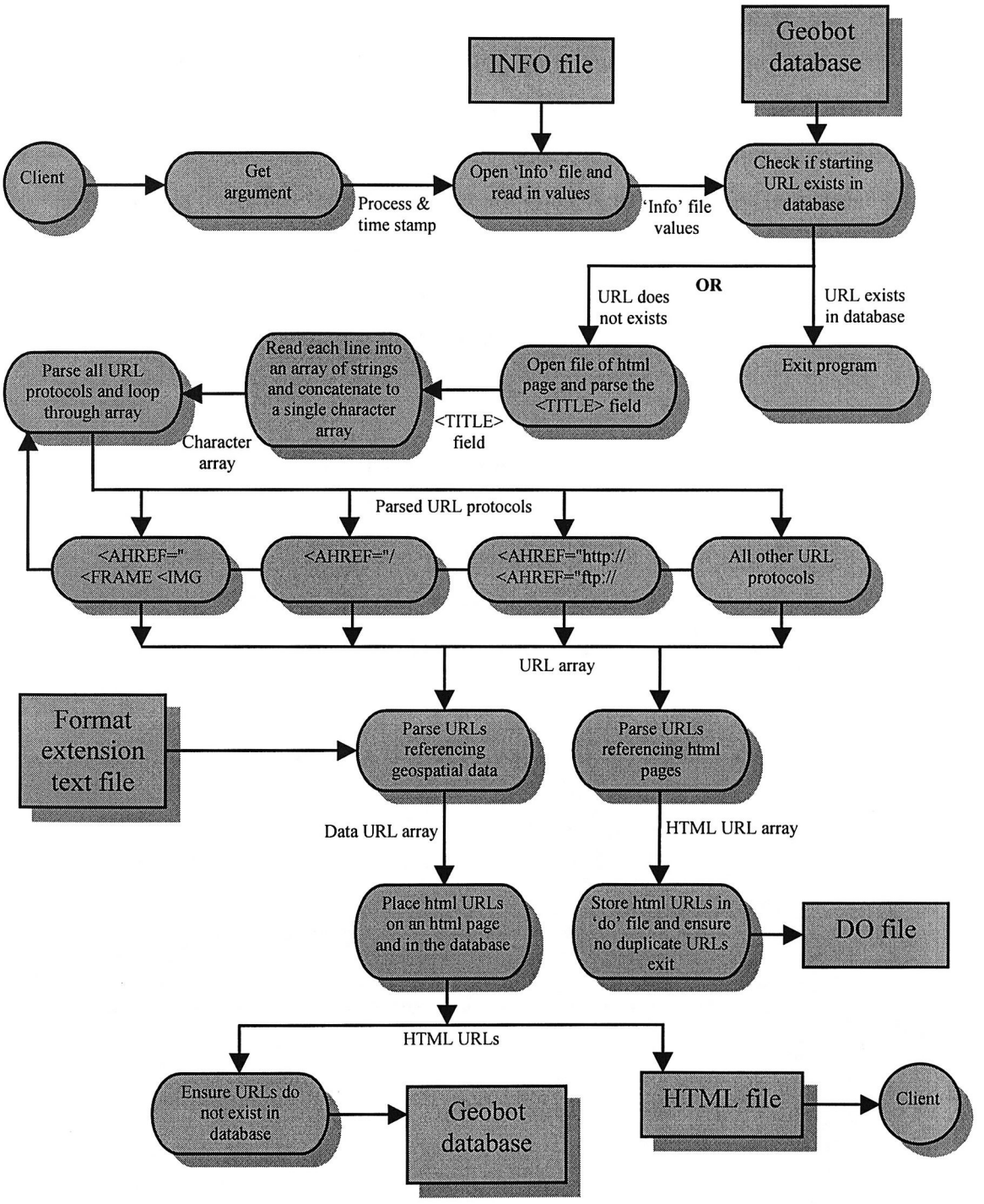


Figure 4.39 Flow chart of the Geobot parsing algorithm. Note the robot's ability to identify HTML frame pages, relative URLs, HTTP/FPT URLs and other Internet protocols.

4.3.2 Software implementation

The implementation of Geobot consists of a detailed description of the various components that comprise this utility. Samples of the HTML pages that form the user-interface are presented and their individual components identified and explained. The operation of the database query and maintenance algorithms is presented. This describes the utilities that allow users to query the Geobot database and remove old or redundant links. The last two sub-sections provide a detailed description of the "crawling" and "parsing" algorithms that allows the robot to move between Web pages and examine their content.

4.3.2.1 User interface

Geobot is a server-side utility that is executed by clients over the Web with a user-interface constructed of standard HTML forms. The user-interface consists of four pages that constitute three major components of the utility and a page for demonstrating the software. The page used for the database query algorithm consists of a single text field about sixty characters in length followed by a submit/reset button (Figure 4.40).

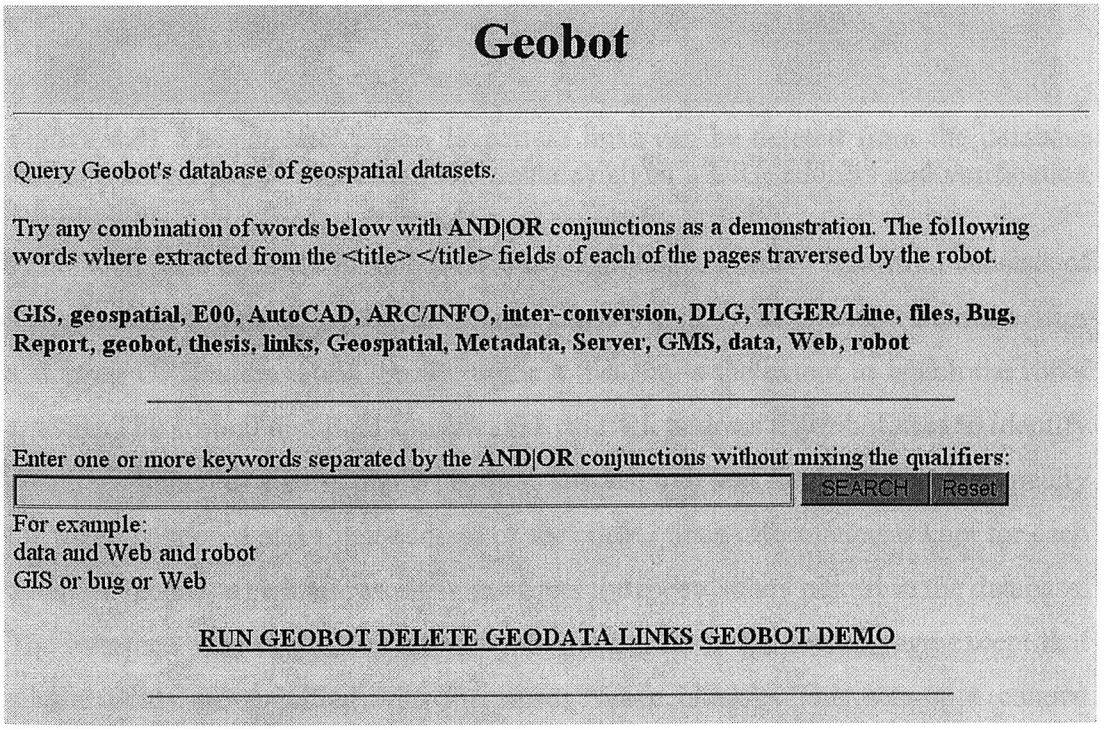


Figure 4.40 The Geobot database query page. It consists of a single field where one or more search-terms or keywords can be entered and joined with Boolean operators.

The Web page used for the database maintenance algorithm consists of a single form composed of three text fields. The form allows the deletion of a page or pages that are previously submitted to the database by the client. The first text field allows the client to specify the URL address of the page or domain (group of pages) to be deleted from the database. The other two text fields, specifying the name/password and email address, are used to match one or more URL addresses within the database to a particular client. The radio button allows the client to either delete a single page or a whole site domain of Web pages from the database (Figure 4.41).

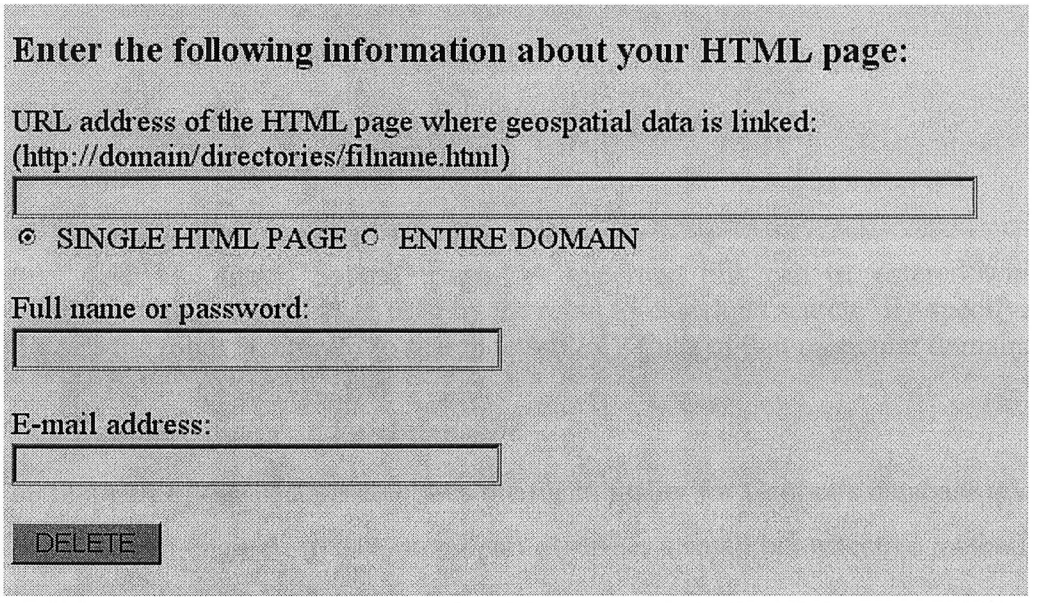


Figure 4.41 The 'Delete' page. Hypertext links can be deleted from the database (either a single page or an entire site domain) given a URL address and verification information.

The Web page used for the robot crawling and database entry algorithm consists of a single form composed of four text fields (Figure 4.42). The first text field specifies a starting URL address and the site domain that limits the extent to which the robot crawls. The second text field specifies the ftp URL address the robot uses to identify resources linked to a particular FTP site. The last two text fields are used to specify the name/password and email address of the client. Such information is kept for each database record, which allows users to access the records they placed in the database. The interface used to demonstrate the robot is similar to the search page except that all the fields are specified with the words 'leave blank'. This sets-up a canned demonstration of the software.

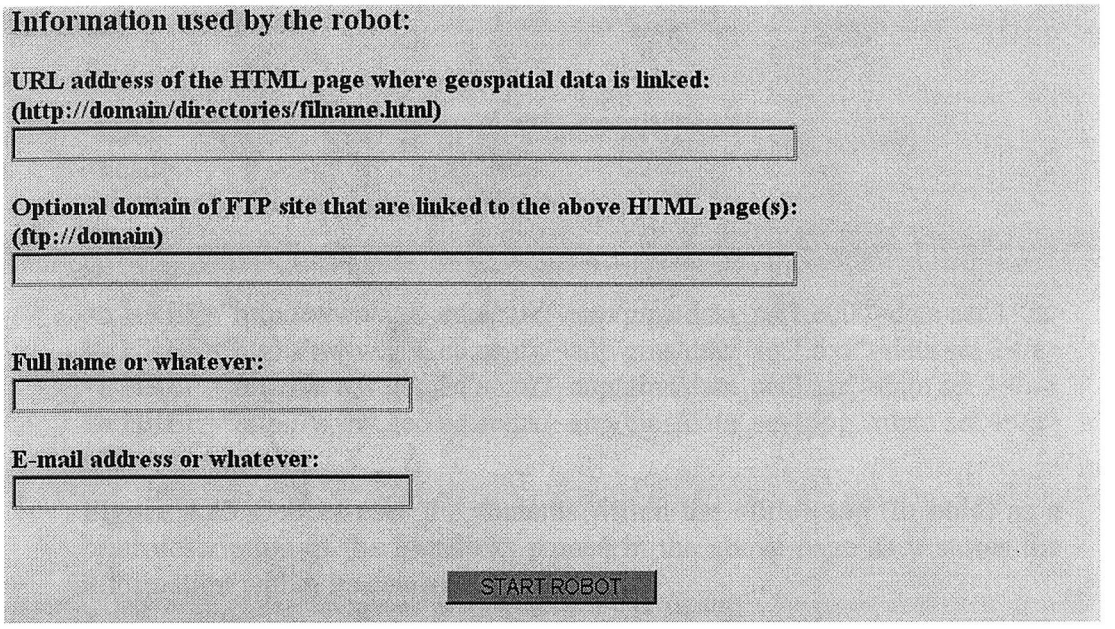


Figure 4.42 The robot 'Submit' page. A hypertext link can be entered which specifies a starting HTML page used by the robot to launch its search. By specifying a FTP site the robot is directed to search for all FTP links of that particular domain.

4.3.2.2 Database query and maintenance

The first part of this section describes the functionality for Geobot's database query utility or search-engine and the second part describes a small but effective method to maintain the database and ensure that superfluous records are removed. The user controls both of these components of Geobot.

Geobot is a fully functional Web robot capable of accepting URL addresses entered by the user and searching an entire site domain for hypertext links to geospatial data sets. Users can release the robot on their own Web sites, enters links to their geospatial data sets into Geobot's database which are then made accessible via Geobot's search-engine. Alternatively the robot can be released on a remote Web site, which then compiles a text file of links to geospatial data sets that can later be viewed by the user. The robot collects a considerable amount of information while searching a remote Web server. This information is in the form of hyperlinks to geospatial data sets that are stored within the robot's database as individual records. Each record also stores some descriptive information regarding the data set, which includes the contents of the HTML title tag from each traversed Web page. For example, the robot stores the URL addresses for data sets 1 through 4 in four

different database records with the title 'Geospatial data sets' as descriptive information. See below:

```
<html>
<head>
  <title>Geospatial data sets</title>
</head>
<body>
<a HREF="http://www.geo.ed.ac.uk/~anp/gms/data_set1.e00">data set 1</a>
<a HREF="http://www.geo.ed.ac.uk/~anp/gms/data_set2.dxf">data set 2</a>
<a HREF="http://www.geo.ed.ac.uk/~anp/gms/data_set3.tgr">data set 3</a>
<a HREF="http://www.geo.ed.ac.uk/~anp/gms/data_set4.jpg">data set 4</a>
</body> </html>
```

Figure 4.43 Geobot uses the contents within the <title> tag (in bold) as a brief description of the four links parsed in the above page as it stores the information within its database.

A query mechanism accessible over the Web allows users to view the contents of the Geobot database. Database records are queried by submitting one or more search words separated by the 'AND' or 'OR' Boolean operators. The words are compared individually with each of the words comprising the descriptive information for each database record. If an exact match is found, then the entire database record is returned to the client. The AND operator requires that all the search words exist within the descriptive information of a record before it is returned to the client. The OR operator requires that at most one of the search words exist within the descriptive information of a record before being returned to the client.

An example of a record returned to the user as a result of a search request is provided below in HTML.

```
13. <a href="http://www.geo.ed.ac.uk/~anp/public/ptestlinks.htm">Anup's thesis related links for the
Geospatial Metadata Server GMS.</A><br><br><H4><a
href="http://www.geo.ed.ac.uk/~anp/linktest /UTM_CITIES.e00">UTM_CITIES.e00</a></H4>
```

The descriptive information for the geospatial data set is linked directly to the page from which it originates; in this case, ptestlinks.htm. The geospatial data set, utm_cities.e00, is linked to its source file. The following table demonstrates how the search-engine operates using the above database record as an example.

Table 4.2 Search-engine demonstration

Search term(s)	Record returned
Geospatial	yes
Conversion	no
Geospatial and conversion	no
Geospatial or conversion	yes
links and geospatial	yes
Conversion or data	no

The database maintenance utility, which comprises the second part of this sub-section, allows users to delete records on one or more of their HTML pages from the database. Since all the database records within the database were originally entered by successive users, a maintenance facility that allows users to remove their obsolete records provides them with a degree of control over the system and what it stores. Database records associated with a single HTML page or an entire site domain can be deleted by specifying the URL address to one of the Web pages in the database and the appropriate username and password. The utility compares the URL address, username and password with each database record. When a match is found, the record is deleted from the database.

For example, links to geospatial data sets are placed on one or more on-line HTML pages that are associated with a particular Web server or site domain. Users who wish to add these links to Geobot's database first run the robot on their Web sites specifying a particular username and password. The robot searches for links within the site domain and stores any that are found in the database along with the links to the HTML pages on which they were found. The username and password that is submitted upon starting the robot is also stored in the database as part of each record. Over time, clients may decide to alter their Web sites by removing existing HTML pages, adding new ones and moving data files between directories. All this activity alters the essential characteristics of a Web site upon which HTML links are maintained. The lack of 'link tracking'[9] is a major problem with the WWW. Under such circumstances, users may want to update the information in Geobot's database that contains links from their Web site's previous configuration. The database

[9] Because Web pages are constantly moved between directories and servers, the HTML links that point to these pages become obsolete. A problem often referred to as 'broken links'.

maintenance utility can be used to delete links from Geobot's database by simply specifying a URL address or site domain, username and password. The software checks each geospatial data link against this information and deletes the appropriate record from the database. The client is provided with the option of either deleting a single HTML page containing data links or an entire site domain comprising of many HTML pages.

As is shown in Figure 4.37, the database maintenance utility is developed in C. The program arguments originate from the HTML form and are processed by the CGI script. The information supplied by the HTML form (Delete Page, see Figure 4.35) is initially stored in a temporary file on the server. The C program is supplied with two command-line arguments by the CGI script, a process/time stamp to identify the temporary file and the page/domain option to specify whether the data links of a single HTML page or entire site domain are to be deleted. The program then reads in the form values from the temporary file, which include the URL address of the HTML page, username and password. If the user selects the 'single HTML page' option from the form, then any geospatial data links within the database matching the URL address are deleted. If the 'entire domain' option is selected, then the utility deletes all the geospatial data links from the database that match the name of the particular domain name.

4.3.2.3 The robot crawling algorithm

The purpose of the robot crawling algorithm is to maintain a list containing URL addresses that are sequentially accessed as the robot operates through a site domain. The HTML pages are downloaded and subsequently parsed of their URL addresses by the parsing utility mentioned below. The CGI script for the crawling algorithm was developed around the Fetch_HTTP utility. This freeware utility essentially releases the developer from having to code using the VMS TCP/IP C library and, in this instance, has considerably cut the development time for this portion of the project. The robot crawling algorithm simply automates the Fetch_HTTP utility so that it can be supplied with its command line arguments while the robot is in operation as opposed to interactively by the user.

The list compiled by the robot is a simple text file containing URL addresses of all the HTML pages that the robot accesses while in operation. This 'DO' file as it is known, provides the robot with its ability to 'crawl' between HTML pages within a site domain. New URLs are constantly added to the 'DO' file by the parsing algorithm as the robot traverses a site domain. As each HTML page is parsed, the URLs extracted are promptly processed and then amended to the list of URLs currently in the 'DO' file. Figure 4.44 further clarifies the process.

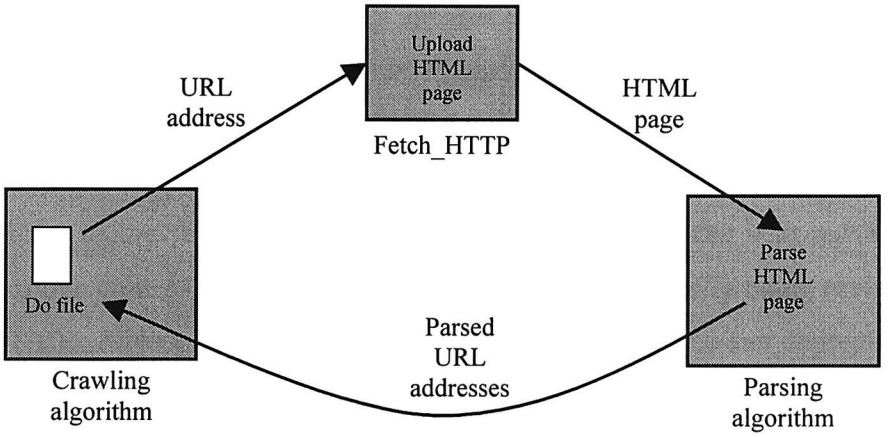


Figure 4.44 Mechanism the robot uses to acquire new URL addresses to traverse a site domain.

The 'DO' file is created at the start of the crawling algorithm, which operates until all the URLs in this file are exhausted.

The crawling algorithm is divided into two sections. The first section begins by supplying the script with four arguments that include the starting HTML page address, ftp address, username and password. The script then extracts the HTTP and FTP domain names (e.g. http://www.geo.ed.ac.uk and ftp://geod68.geo.ed.ac.uk[10]), that are used through the entire operation of the program. The domain name information and the HTML form field arguments are placed in an 'INFO' file and the first entry in the 'DO' file is the address of the starting HTML page. The starting HTML page is uploaded and is verified by the script. In the event of a 'broken link' the Fetch_HTTP utility supplies an HTML error message. The script reads the first few lines of the uploaded HTML page and decides whether to activate the parsing program or to skip the first URL altogether. In either case, an HTML submit button

[10] The Fetch_HTTP utility does not upload ftp documents. Rather the user specifies the ftp domain name and the robot uses it to search for ftp URLs that match only domain name.

is printed to the user's browser that effectively deletes the 'INFO' when pressed and stops the robot. Therefore, the client is given the ability to start and stop the robot thereby reducing server load and bandwidth.

If the starting HTML page is properly uploaded and parsed, then the second section of the program is initiated. The URL addresses parsed from the starting HTML page are placed in the 'DO' file by the parsing algorithm. The script then sequentially loops through all the URLs in the 'DO' until the end of the file is reached. As in the first section, the main loop in the script uploads each HTML page, checks for a 'file not found' error, decides whether or not to parse the page and appends any URLs found by the parsing program to the end of the 'DO' file. When an 'end-of-file' flag appears while the script is reading the 'DO' file, the main loop exits and the program provides links to the 'traverse' and 'links' file (explained in the next sub-section) before returning control to the user. With the 'DO' file continually being replenished with new URL addresses for each iteration of the main loop, one can appreciate that Geobot can potentially operate for a considerably long period of time depending on the number of pages contained in the site domain.

4.3.2.4 The robot parsing/database entry algorithm

The Geobot parse and crawl algorithms closely complement each other through the information transferred between the two programs. The crawling algorithm needs a steady supply of URL addresses from the parsing algorithm in order for it to traverse an entire site domain. The parsing algorithm needs HTML pages uploaded with the Fetch_HTTP utility in order to extract relevant URL links of on-line geospatial data sets and other HTML pages present on the Web site. The links with an HTML file extension are placed in a 'traverse' file that lists the Web pages visited by the robot. The links that reference on-line geospatial data sets (file extension other than an HTML file) are placed in a 'links' file, which conveniently compiles the information required by the user to access the resources. The detailed process of parsing an HTML page is shown in Figure 4.39.

After the HTML page has been uploaded and saved to the local system using a process/time stamp for a file name, the first task of the parsing algorithm is to

determine whether or not the URL address referencing the page exists in the database. The following Pro*C code is used to accomplish this task:

```
select GIS_ROBOT.GS_ID, GIS_ROBOT.HTMLURL from ops$anp.gis_robot
    where upper(gis_robot.HTMLURL) = upper(:vchtmlurl);
```

The table name is GIS_ROBOT; the two columns that are queried include the record number, GS_ID, and the URL address of the on-line HTML pages, HTMLURL. Here, HTMLURL is being compared to URL references stored in the 'INFO' file, see Figure 4.39. If the URL does exist in the database then the parsing program is exited and the crawling algorithm reads and uploads the next URL from the 'DO' file. This process is to ensure the database is not cluttered with duplicate records. If the URL does not exist in the database then the <TITLE> field of the HTML page is parsed to provide a general description for the geospatial data links that are found in this document.

Such descriptive information about each geospatial data set complements the rest of the information stored in the database. The following table outlines the nature of the information in the database.

Table 4.3 The Geobot database

	gs_id	htmlurl	title	Fullname	email	domain	filename	fileurl
Row #								

The primary key labeled as 'gs_id' for the above table is created using ORACLE's database sequence generator. The 'fullname', 'email' and 'domain' columns store the user name, e-mail address and domain name; all three values are taken from the 'INFO' file. The values for the last two columns 'filename' and 'fileurl' are derived from the URL address of each HTML page.

The parsing algorithm initially reads each line of the HTML document into individual character arrays and concatenates each array into a single very large character array. All the blank spaces in the document are removed from the array since such spaces within HTML documents do not impart any functionality but are there more for the purpose of clarity. The resulting large character array is then placed in a loop where the characters comprising the array are advanced a single

character at a time. The document is parsed as the characters are advanced throughout the entire document.

The main parsing loop acts very much like a sieve by successively extracting URL protocol extensions followed by the required HTTP and FTP domain names. URL addresses within HTML documents contain anchor elements <A>...</A> which are equated to the 'HREF=' attribute providing the target of the hypertext links. It is this target that forms the basis of parsing URL addresses from HTML documents.

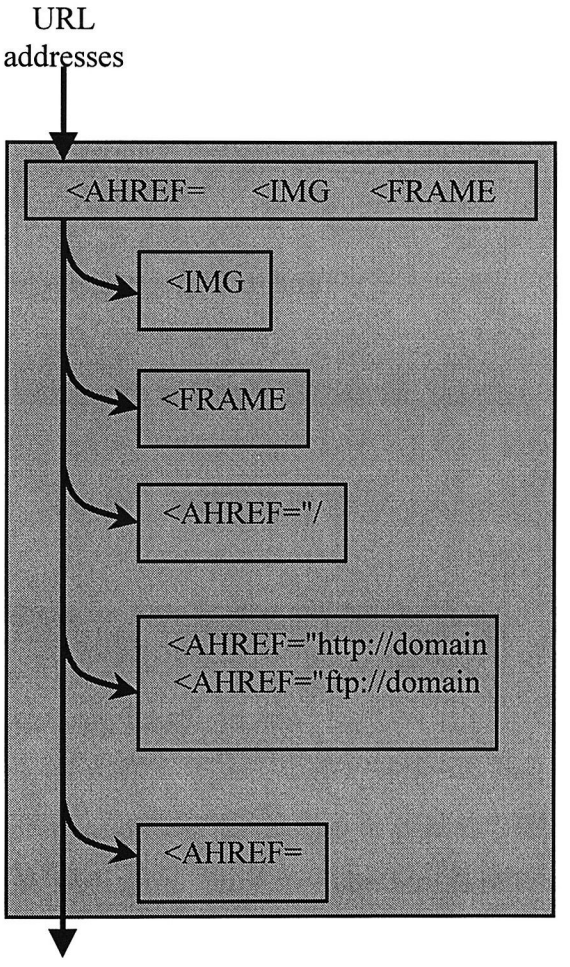


Figure 4.45 The 'sieve' like parsing mechanism for Geobot. Hypertext links are individually examined by placing the relevant ones in separate C language arrays for further use later in the program.

The robot identifies the most common URL protocols by looking for a series of attribute tags with <AHREF being the most commonly sought after. The inline

image elements (<IMG) are also singled out because a considerable amount of graphical information is linked through the attribute. Also, many of today's browsers allow multiple independent-document viewing panes that are designated by the <FRAME element. Once the parsing algorithm identifies the inline image and frame elements, their contents (file name and URL address) are extracted using separate methods as shown in Figure 4.45.

The robot must then identify four domain-specific URL types. Two of these are known as partial URLs. The <AHREF= reference tag indicates a hypertext link with the same domain and directory path as the document from which it is linked. For example, given the following URL which contains a link that references a page called form.htm:

http://www.geo.ed.ac.uk/~anp/gms/search.htm

The partial URL for form.htm as written in HTML is:

<AHREF="form.htm"

However, the URL address for form.htm would still be:

http://www.geo.ed.ac.uk/~anp/gms/form.htm

Similarly, directory information can be placed in partial URLs without having to repeat the domain name each time. So if the form.htm is referenced from the home page:

http://www.geo.ed.ac.uk/~anp/home.htm

It can be represented in HTML as:

<AHREF="/gms/form.htm"

The client defines the remaining two URL address types when the robot is started. All other URL addresses that include those associated with other HTTP or FTP sites as well as other Internet protocols are discarded by the main parsing loop.

The URL addresses parsed from the HTML page are placed in an array for use in the sorting portion of the algorithm. The resulting URL array contains addresses to both geospatial data sets and HTML pages from the Web server that the robot is instructed to search. Recall that the geospatial data links are placed in the database. The HTML links are placed in the 'DO' file ensuring that there is not repetition. Repetition of URL addresses within a robot's traversing mechanism would cause the software to loop infinitely

The robot examines the file extensions of the documents to identify the geospatial data links from the HTML links. This is achieved by examining the last one to five characters of each link in the URL array and sorting each URL address in either of the two categories. HTML documents are recognised with a '.htm' or '.html' file extension and placed in an array called 'HTML URL array'. The file extensions recognised by the Web robot for each of the geospatial data links are placed in another array called 'DATA URL array'; they are listed and defined on the CD-ROM in appendix A.

The final stage of the parsing program is to ensure that duplicate URL addresses in the HTML URL array are removed. Each URL address is also checked against the ones that already exist in the 'DO' to ensure no duplication exists. Once the parsing is complete, the program control is returned back to the crawling algorithm.

4.3.3 Design and implementation constraints

Geobot is one of the few Web robots that can be executed by any user of a Web browser. The prospect of having multiple concurrent processes of this robot running on just a single local server is a real possibility. When Geobot was first tested on its local server (domain name www.geo.ed.ac.uk), it was found to download and parse some 672 pages in just under 30 minutes. If many other users repeated this same test, the server load on the local server would be considerable because (like most

Web robots), Geobot is an automated tool requiring vast amounts of network resources. The test was conducted on the server that contains the robot, so bandwidth considerations need not be taken into account. However, widespread adoption of such tools by the general public with their possible use on any Web site puts considerable strain on network resources.

4.4 Chapter conclusion

This chapter examines three of the software components that comprise the Geospatial Metadata Server devoted towards the on-line query and analysis of metadata. The on-line GMS data-conversion utilities provide a powerful means by which the GIS user community can convert between different geospatial data formats without having to incur the expense of purchasing and implementing additional software. Interchange standards address the pervasive problem of GIS interoperability as a consequence of the incompatibility of vendor-specific data formats that are now widely available to GIS users over the Internet. The development of flexible and comprehensive search mechanisms along with an intuitive user-interface is of considerable importance in querying on-line metadata and geospatial data. There are a variety of tools to construct search-engines for the Web, which are capable of querying geospatial data and its metadata. Though GMS is capable of performing relatively sophisticated date, area and keyword queries, an interactive graphical user-interface is required for the search-engine in order to achieve the simplicity required by today's Web users.

Indeed, on-line query and analysis tools via Web search-engines are only an initial step towards developing far more powerful tools to search the Internet for geographic information. Interactive surrogate softbots or Web robots charged with finding on-line geospatial data are a relatively new phenomenon because they accumulate metadata information from the Web which can be browsed by clients or used within on-line GIS software. A Web robot such as Geobot is designed to catalogue automatically the URL addresses of metadata files, geospatial data sets, associated graphic files and other geographic information. The use of Web robots in GIS is relatively new topic of research requiring further study.

CHAPTER FIVE

DISCUSSION AND EVALUATION

This chapter focuses on how developments in the computing industry directly affect the direction of future GIS software development. The rapid adoption of client/server technology (as it applies to the WWW) impacts the computing industry as a whole. Since the Internet is increasingly being adopted as a new computing architecture for GIS, its principal components including the storage, analysis and display of geospatial data can operate at physically remote locations from one another over a distributed network. One of the primary objectives of this thesis is the development of a prototype system consisting of six dependent software components that demonstrate how geospatial data and its metadata can be used over the WWW. A further thesis objective is to evaluate the procedures and problems highlighted in chapter one by the implementation of each component and suggest strategies for their improvement and further development. The sections contained in this chapter discuss a range of different topics, which includes software implementation, performance, evaluation and evolution.

The method used to evaluate each of the GMS components were outlined at the beginning of chapters three and four. The extent of the human evaluation component of this thesis consists of eliciting electronic-mail responses from people who use one or more of the GMS on-line utilities. Users of the system are provided with a brief on-line description of each GMS component in order to facilitate a basic understanding of the purpose, design and operation of the software. Sample geospatial data sets, metadata files and search-terms are provided to users wanting to test easily each GMS component. As each GMS component was completed, their use was promulgated on GIS related Usenet newsgroups as well as the GISTRANS listserver, which is a server devoted to issues of translating or converting geospatial data from one format to another. The entire system was also listed on the

Department of Geography's WWW list of GIS related Web sites.

These methods elicited many responses from geospatial data users, providers and site administrators. However, approximately only one hundred and fifty responses were compiled in the evaluation of GMS. These responses vary from problems associated with the operation of the software, suggestions on its further improvement, comments on the effectiveness of a particular component as well as general comments on how such utilities affect the future use of geospatial data over the Web. The advantage of using this passive method of evaluating on-line software is that the people used in its testing are genuinely interested in issues related to cataloguing metadata, on-line database searching, data conversion and/or specialised software agents. In many cases, they are at the forefront of their fields with respect to issues related to the on-line distribution and analysis of metadata and geospatial data.

5.1 Metadata parsing and database entry

The metadata database forms the core of GMS allowing various system components to operate with each other through the exchange of information. The CSDGM standard is the primary means within the metadatabase to describe geospatial data. The standard exhibits a broad definition of conformance by consigning the structure and layout of metadata files to their creators (Schweitzer, 1996a). Even though terminology may be consistent, this lack of structural uniformity of CSDGM compliant metadata files often precludes their use within on-line parsing programs and makes storage difficult within relational databases.

"It would be quite logical for *mp* to generate an output format for RDBMSs. The problem I see is that the table names tend to vary and the table structure may vary as well, from one DBMS implementation to another. The local schema I develop will be different from the one you develop, not just in the names of tables and fields, but also in which fields we choose to include."

Peter N. Schweitzer
U.S. Geological Survey

The metadata-parsing program known as *mp* was reviewed in section 2.3 of the

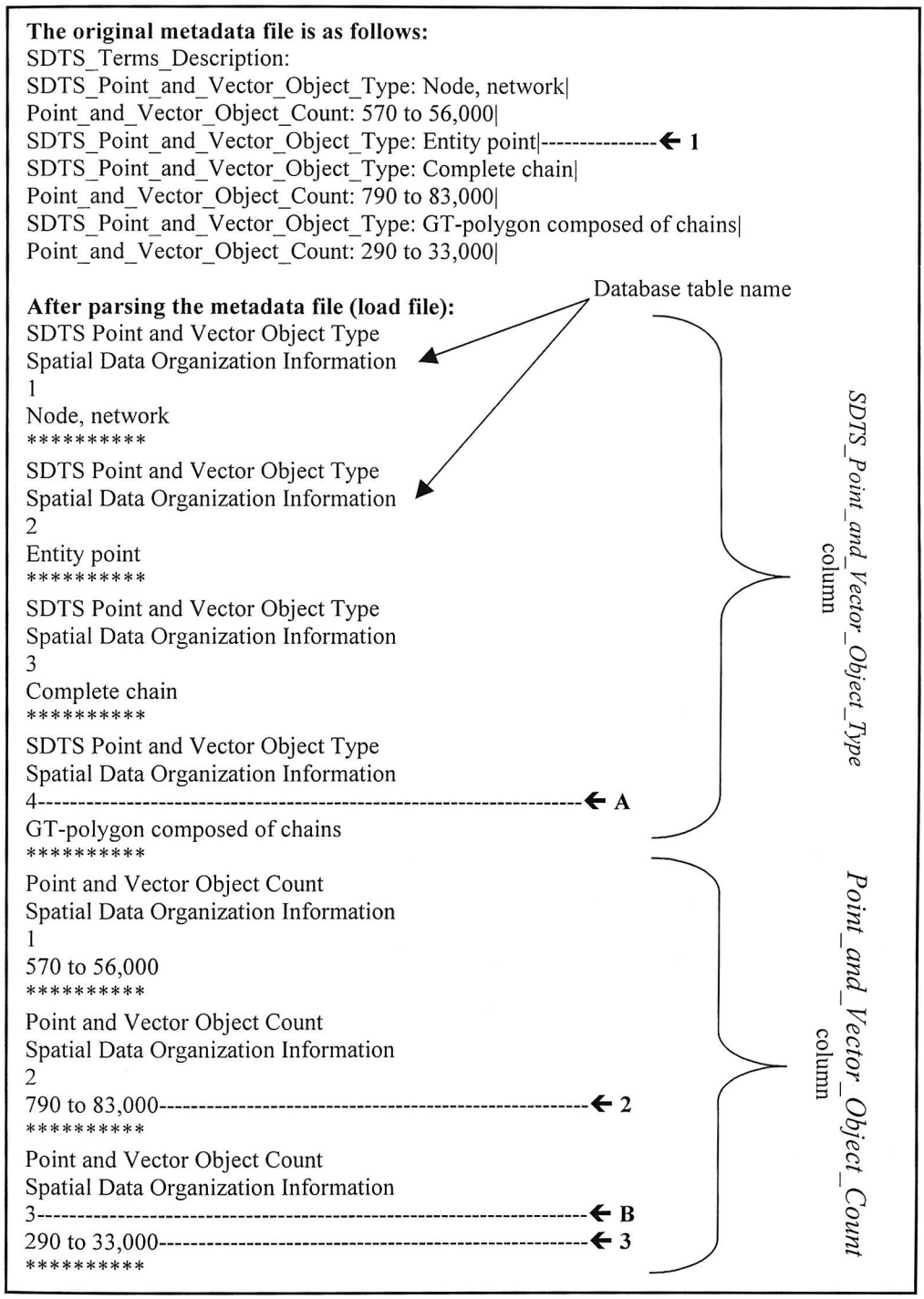
second chapter. The exchange of metadata is often impractical between different on-line system components, thereby preventing their collective use over the Web unless some form of standardisation between database schemas is achieved. Also, if metadata is unable to be parsed of relevant information, then it is difficult for site-administrators to maintain the currency, accuracy and quality of metadata entering their on-line databases. It is also difficult to build-up such databases unless data-providers are given a quick and easy means to enter their metadata files into a centralised metadata catalogue.

The GMS metadatabase successfully proposes a standardised database schema for a portion of the CSDGM standard that can be used by one or more site developers to store compliant metadata as database records. The database schema is not a full implementation of the standard consisting of fourteen tables and more than eighty elements. However, the number of represented elements is enough to demonstrate the full operation of the database by supplying metadata to a system of on-line software utilities. Developers of on-line geospatial data dissemination sites require the full database implementations of their chosen metadata standard. Because of the size and complexity of geospatial data sets, developers want to implement an entire metadata standard that incorporates all aspects of currently produced geospatial data for that standard. The GMS metadatabase is only a partial implementation of the CSDGM standard. The development of a database schema that covers the entire CSDGM standard and its implementation on more than one database server is the next step in developing federated systems for metadata and geospatial data distribution.

The metadata parsing and database entry utilities (MPARSER and MLOADER) successfully parses complex CSDGM metadata files for section names, element names and element values and inserts the information in the proper tables and columns of the GMS metadatabase. Both utilities allow providers of geospatial data to upload their metadata files and subsequently parse and enter them into the database. Such functionality is of considerable benefit to site-administrators for reasons mentioned above. However, users of this software indicate these utilities are not only difficult to use but also somewhat unpredictable. Each metadata element and element value parsed by the MPARSER program must be properly delimited.

This requires the data-provider to have an in-depth knowledge of the CSDGM standard including the distinction between mandatory versus optional elements and their relationships with different sections of the standard. If the information is incorrectly placed in the metadata file, then the MLOADER program is unable to entry the information into the database due to problems associated with the integrity of references among database tables, improper data types or out of range values.

The MPARSER metadata parsing utility is used to parse CSDGM metadata files for entry into the metadatabase. Chapter 3 focuses on some of the advantages of using an on-line metadata parsing utility over various types of standalone parsing programs. In this context, there are a number of techniques to improve an on-line parsing program such as MPARSER. One such improvement is by reducing or eliminating the effort required in delimiting metadata files prior to sending them through the MPARSER program. The parsing program requires substantial modification in order to remove the need to delimit each metadata value within a metadata file. Another improvement is to expand the program to incorporate other metadata standards used in the GIS industry. This makes the utility more versatile and relevant to a wider number of users. However, among the most common problems with this utility while parsing relatively complex metadata documents is in assigning the OCCURRENCE values to a one-to-many relationship. Sometimes the columns do not line up with their proper data values. The example below shows the problem.



The above parsing exercise prepares the values for the *SDTS_Point_and_Vector_Object_Type* and *Point_and_Vector_Object_Count* columns of the *Spatial_Data_Organisation_Information* metadatabase table. Notice there are no values for 'Entity point' (arrow labelled 1) in the metadata file but they

are given the values '790 to 83,000' (arrow labelled 2) after parsing. In fact in the *Point_and_Vector_Object_Count* column, 'Entity point' is assigned the value of 'Complete Chain' (arrow labelled 2) after parsing and likewise, 'Complete Chain' is assigned the value of 'GT-polygon composed of chains' (arrow labelled 3). Also, there are four rows in assigned values of object types in the *SDTS_Point_and_Vector_Object_Type* column (arrow labelled A) and only three rows assigned values of object count in the *Point_and_Vector_Object_Count* column (arrow labelled B). Both columns should have an equal number of rows because they are both part of the same database table (*Spatial_Data_Organisation_Information)* and store information for a single database record.

The problem described above shows that parsing programs are fallible and that the error can be extremely difficult to detect given the size of a typical metadata file and the amount of metadata data that is parsed. So in the case of the GMS parsing program, it is essential that the user check the row/data values of all the one-to-many and many-to-many relationships within the load file. Or alternatively, the user can enter mock values for fields in the metadata file so that the row and column values line up after the document is parsed. This is only a temporary solution for the problem associated with the program in question. Something more permanent envisages a parsing utility capable verifying the information written to the output file. Additional processing invariably leads to a decrease in performance, which can be particularity noticeable to the client when implemented as part of other Internet applications.

Parsing programs, in general, experience various problems associated in the accuracy with which they parse a metadata file, particularly those that operate on complex metadata standards. In many respects, the errors are unavoidable unless stringent measures are taken comparing each element and element value within the metadata file with its corresponding value in the parsed file. The loose criteria on conformance associated with CSDGM metadata files renders them inherently difficult to parse because of the large variations associated with the format of these documents. In essence, there may be uniformity of syntax but formatting such as line spacing and indentation as well as the use of nonessential characters can deceive

and even fail the most robust parsing programs.

While testing the MPARSER program as well as through feedback from users who managed to parse their own metadata files or ones they found on the Internet, it is evident that the parsing utility does not operate effectively with certain formats of CSDGM metadata files. For instance, there is a growing amount of standardised metadata on Web in HTML format. Many data-providers may find it easier to produce such documents using common HTML editors that allow their file to be subsequently placed on a Web site. Upon parsing such metadata files, the program can not distinguish between the element values and the HTML tags used to format the document for the Web. In fact, the utility merges the values and the tags into a single string resulting in element values with extraneous character on either side. One can view the MPARSER output file as an HTML document within a Web browser but all of their extraneous characters eventually find there way into the metadatabase via the MLOADER program. This, in turn, creates problems for other GMS components such as the search-engine and Web robot, which draw upon information contained within the database. Originally, MPARSER was designed to parse only flat text files of varying formats consisting of differing line spacing and levels of indentation. In this respect it is fairly competent. However, in order for the program to process a wider variety of file formats currently available on the Web, its algorithm needs to be modified in order to identify and remove HTML tags or allow the data-providers a choice between parsing text or HTML file formats.

The MLOADER program can also be improved because at times it behaves rather unpredictably due to the referential integrity constraints placed upon the database. Even a modest database representation of the CSDGM standard can be relatively complex. The software built around this database, in particular the programs that insert and retrieve information from various tables and columns, must be able to effectively manage this complexity by reconciling all the relationships between the database tables. MLOADER is a series of programs that inserts metadata in a database from a pre-processed metadata file. Each program must be able to insert information in the rows and columns of tables following a specific order because of the nature of the relationship that exists between the database tables (see chapter 3.1). If there is a problem in even a single database entry program then the entire process

is aborted and any changes made to the database during the operation are reversed using the SQL ROLLBACK commands.

By providing the user with more error feedback and control over the operation of each database entry program, some of the problems associated with entering metadata into the proper tables and columns of the metadatabase are significantly reduced. Once executed, stopping and re-starting the various programs that comprise the MLOADER utility is not possible because HTTP establishes only one connection to the server for each request. Alternatively, the programs can be engineered to operate independently by acting upon separate requests from the client. However the programs must operate in a specific order so that the client can monitor the success or failure of inserting metadata information into each database table as well as avoid problems that may arise with breaking any referential integrity rules. In any event, automatically inserting metadata into a relational database requires sophisticated functionality and a feature rich Web interface to control the entire operation.

Finally, the GMS metadatabase is designed to operate as a single-user database where the user can modify data within a transaction without concern of other users modifying the same data at the same time. As mentioned previously, individual rows within the metadatabase contain no more than a single metadata record, which is associated with a single author who is identified by a personalised username and password. So at any time, the data contained within the table and column of a particular row can be accessed by no more than a single user. The on-line metadata editing utility (discussed in the following section) operates under this principle. Each operation, executed from a Web client, on a particular row within the database must be completed before another operation can take place. If the username and password has not been compromised, only a single on-line user can access a single database row at any particular time.

In the event of a compromised authentication code, problems arise with data concurrency and consistency in a multi-user database situation where simultaneous multiple transactions can update the same data. The ORACLE DBMS used for GMS ensures that concurrently executing transactions produce the same effect as those serially by a single user. The database ensures a consistent view of all the data committed by other transaction as well as all the changes made by the user up to a

point in time. For example, data returned from a query is inconsistent when changes of another committed or uncommitted transaction are visible at the time a particular transaction takes place. At times a particular database query can not be repeated because one or more other users are altering the data by inserting new information, updating existing information or deleting tables or columns of a particular row. Updates can be lost when an update of one transaction overwrites an update of another transaction before the second transaction commits. Similar problems of lost information occur if one user drops a table while being updated by another user. These problems are relatively rare while inserting, updating and deleting information from the GMS metadatabase. However, GMS can minimise the possibility of compromising user authentication codes by ensuring each of these codes are only associated with certain IP addresses identified by the metadata author before accessing the system.

5.2 On-line metadata editing utility

The on-line metadata editor developed for GMS allows data-providers to create new, update existing and delete old metadata records using a series of HTML forms each designed to access a metadatabase table. The utility successfully allows each data-provider (or metadata creator) to edit their own metadata records, which are stored using a centralised database server accessible via a Web interface. Such functionality is significant because it absolves site-administrators of the seemingly impossible task of individually modifying a database that can contain thousands of metadata records. Given the length and complexity of typical metadata records, an on-line metadata editing system that allows users to modify their own records is an effective tool for managing centralised or distributed databases over the Web.

The remainder of this section describes some of the implementation problems encountered during software development and user evaluations of the GMS on-line metadatabase editing utility. The main implementation problem encountered with the utility is that its DCL variables or symbols used throughout can not handle values greater than 255 characters in length. The values of the DCL symbols are assigned

their values from CGI environment variables, which are brought over from the clients when the HTTP server receives the 'POST' method invocation through a socket connection. If the values exceed this limit within DCL programs, then the entire field reverts to a null value, which means that the metadata typed into HTML form fields such as Abstract and Purpose are lost when they are used by the CGI script. Since DCL symbols are truncated to 255 characters in length, an alternative method is used to handle metadata entered into HTML TEXTAREA fields with textual length that exceed this limit. The problem can be addressed by using a programming language such as C or C++. Compared to DCL, these lower level languages are considerably more flexible in assigning memory to string variables capable of handling very large amounts of textual information. The implication is that the entire contents of each HTML TEXTAREA field can be sent to the server via the 'POST' method invocation and read directly from the socket connection using a C or C++ program.

Another major impediment in using the on-line metadata entry utility is the need to constantly retype the fullname and password each time a SELECT or INSERT operations is performed from one of the HTML forms. User evaluations confirm that continual entry of verification information is rather cumbersome when performing a large number of database operations. The problem can be addressed by modifying the CGI scripts to store the verification information along with the user's IP address (taken from the environment variables) and a time stamp within an access file that contains similar information of all the users currently accessing the system. Each time a user inserts, selects, updates or deletes rows from the metadatabase, the access file is checked for the user's IP address. Alongside the IP address is the user's verification information and the session time, which can be used to access the user's database records. Even users located at the same IP address possess different verification information and session times. Clients can either manually exit the system or be logged out automatically after some pre-specified time of inactive use. In this way, the database entry and modification utility does not require users to continually enter verification information each time the database is accessed.

On-line users of the GMS metadata entry and modification utility also cite the lack of description for each metadata element. Short textual descriptions and hypertext

links to the CSDGM standard are not enough to convey exactly the sort of information the user is required to enter into a particular field. Detailed explanations of each metadata element including examples are cited as a possible improvement over the existing implementation. Users also have difficulty in visualising the linkages between the various HTML forms that correspond to specific metadatabase tables (see Figure 3.1 in chapter three). The relationships between database tables are rather complex. This complexity is not adequately conveyed to the user in simple layman's terms. For example, rows in the *Identification_Information* and the *Source_Information* tables must each be related to a single row in the *Citation_Information* table; but a single row in the *Idenfication_Information* table can be associated with more than one row from the *Source_Information* table. Given the confusing circumstances, additional information needs to be augmented to the instructions that are currently used for this utility.

Finally, performance studies indicate the metadata entry forms operate with a good degree of speed, due to the fact that most of the processing is done by Pro*C programs. HTML forms with large TEXTAREA fields operate somewhat slower because the script prior to database entry must parse the text before being read by the metadata editing software. Overall, the on-line metadata entry and modification utility is implemented with a considerable degree of success because of the satisfactory operation of the database schema, the HTML metadata entry forms and their associated scripts.

5.3 Geospatial data-conversion utilities

The three data-conversion utilities convert geospatial data over the Internet in E00 format to and from DXF, DLG and TIGER/Line format. Using FTP and HTTP to upload files onto the local server, the E00 interchange format can be used as a common data format allowing on-line conversion between the other three data formats. The utilities successfully demonstrate that such operations can be performed over the Internet allowing users of geospatial data who have acquired data in a format they can not use to convert the data into a format that can be used within

their own GIS systems. However, the implementation of these conversion utilities reveals interoperability problems associated with the differences in the representation of information contained in files created by different operating systems. The suitability of other conversion programs currently available on-line or as standalone software is assessed alongside the conversion programs used for these on-line utilities. Finally, the operation of the GMS conversion utilities are evaluated with user feedback, which identifies specific implementation problems followed by proposals for suitable changes in design.

Converting geospatial data over the Web provides the user platform-independence in the operation of GIS software. The user need not choose between the conversion commands supplied by different GIS packages. The widespread use of geospatial data is largely dictated by the degree of interoperability between different operating systems running GIS software, versions of the same GIS software and GIS software packages. The number of supported spatial data formats varies between different GIS packages. Each GIS package supplies only certain commands to convert spatial data to and from its native format. The conversion commands present within the system usually include formats produced by various government organisations as well as some interchange formats. However, the list of supported geospatial data formats is rarely exhaustive because many data formats are excluded from various industry and government organisations. Also, new data formats are continually created forcing the users into relatively complicated and expensive software upgrades. For example, the list below contains the Arc/Info version 6 data-conversion commands used to translate data from Arc/Info coverages to and from the following formats (ESRI, 1991):

- Automated Mapping System (AMS)
- Dual Independent Map Encoding (DIME)
- Digital Line Graph (DLG)
- Auto CAD Drawing Interchange File (DXF)
- ETAK MapBase file
- Land Use and Land Cover Data (GIRAS)
- Interactive Graphic Design Software (IGDS)
- Initial Graphics Exchange Standard (IGES)

- Map Information Assembly Display (MIADS)
- MOSS export file
- TIGER/line

Notice the absence of other established GIS formats such as Map Info, Autodesk, Genasys, Smallworld, etc. as well as the data formats produced by organisations of other governments. The previously mentioned interchange standards such as SDTS, SAIF, NTF and CDF (see appendix B) are also absent in this particular version of Arc/Info.

The on-line geospatial data-conversion programs offer a number of key advantages over numerous data converters or translators, which are either supplied as part of GIS software packages or downloaded as freeware from the Web. It is conceivable to have a range of translators (e.g. FME, Arc/Info) as part of a system providing a series of inter-conversions between formats using a common data format. However, the use of Web conversion utilities frees users from the time and expense of installing and maintaining the software on their own systems. Such services are centrally maintained providing convenient and cost-effective on-line data-conversion. A number of freeware conversion programs obtained from the Web were examined for use within the GMS conversion utilities. However, the Arc/Info data-conversion commands, outlined above, were found to be the most practical and convenient means to develop the on-line conversion utilities. Since Arc/Info already forms a stable and powerful back-end GIS application for GMS, it is a relatively simple task to tie the various conversion commands together to form the data-switch-yard.

However, over the long term, vendor-supplied data-conversion utilities possess some major drawbacks over those that operate over the Web. Vendor-supplied GIS software must be continually updated to incorporate any new spatial data formats created since the release of the latest version. Though some software updates can be downloaded from the vendor's Web site to operate with the existing version of their software, many vendors simply require their users to wait till the next version is shipped. Upgrading is potentially very expensive, particularly if the user requires only a small addition in functionality from the present to the latest version of the product. Organisations must also decide if this added expenditure is justified given that it may only benefit a small number of their users.

An alternative to using relatively expensive back-end applications to develop these conversion utilities and provide on-line data-conversion is by downloading freeware geospatial data-conversion software and integrating it with the Web site's existing CGI scripts. For example, Katz (1995) provides software written in ANSI C that converts various spatial data formats to and from the SDTS format. Recall that SDTS is a geospatial data interchange format (see appendix B). Information contained in SDTS format can be converted (or dumped) into ASCII format using the FIPS123 software library developed by SDTS Task Force (Fegeas et al., 1992). Both the conversion programs and the SDTS FIPS 123 software library were originally written for the UNIX platforms. The problem in using the FIPS 123 software library is that it only compiles for certain flavours of UNIX and must be modified for use on other platforms such as the VMS or NT operating systems.

However, the main issue is not the common data format used to perform on-line inter-conversions between different geospatial data formats. This is more an issue of development. It is that on-line systems for converting geospatial data between different formats, such as the one implemented for GMS, need to provide data in whatever in a multitude of formats. For example, geospatial data users do not want data in SDTS or NTF because there are very few software applications that support these interchange formats (Katz, 1995). Most people still prefer data in E00, MGE or DXF formats because it is these and other formats that control most of the market for GIS software.

Whether on-line data-conversion utilities use freeware or proprietary conversion programs as their back-end applications for converting geospatial data, the facilities used to upload this data to the local server is an important system component. User evaluation of the FTP upload program (see appendix F) used in all the conversion utilities revealed that subtle differences in the operation of FTP servers seriously affected the upload of files from these servers. The GMS conversion utilities are developed and tested using geospatial data and metadata files retrieved from the US Bureau of Land Management FTP site at ftp.blm.gov. The original implementation of the FTP upload program accesses the remote files by traversing the entire directory tree a single command line at a time separated by a carriage return.

cd dir1<RT>

cd dir2<RT>

get filename

User evaluation of the E00/DXF conversion utility revealed problems arising with uploading files from certain FTP servers. Downloading files from these servers tended to fail before writing an error message.

"The current procedure you are using to retrieve FTP files from our server returns an error message. I suggest you try using a single command incorporating all the directory paths as well as the filename."

Debbie Anderson
Data-conversion
ESRI-Redlands

The modifications to the FTP upload program incorporated this advice by replacing the directory tree above with a single command line,

get dir1/dir2/filename

This is the default, which is accessible by using the FTP1 radio button, for uploading files via FTP from remote sites (see Figure 5.1). In retrospect, even though there is an adjacent link explaining the exact function of FTP1 and FTP2 radio buttons, the use of such labels tend to make the interface rather ambiguous. User evaluation of the FTP upload utility confirms that a more meaningful label for these radio buttons enhances the clarity of this portion of the interface.

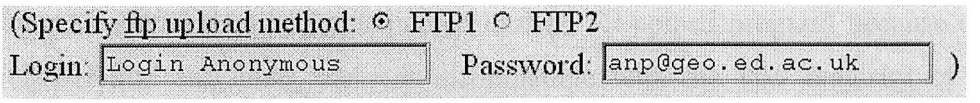


Figure 5.1 Radio buttons and HTML text fields were added to facilitate the access of on-line files located on FTP servers. Each radio button provides a different format for the 'GET file' command. The text fields allow users to enter login and password information to access files from their own servers.

Further testing revealed that the FTP1 implementation did not work with certain other FTP servers forcing the retention of the initial implementation labelled as

FTP2. In essence, the GMS conversion utilities required the implementation of two separate sets of FTP upload programs. Some ftp servers (such as the one used by the person providing the feedback paraphrased above) are not able to handle the anonymous login command when executed from the CGI script of the GMS conversion utilities as follows:

```
ftp ftp.esri.com
login anonymous
anp@geo.ed.ac.uk
ascii
get pub/deba/dutm_cities.dxf  3040e45a_741.dxf
quit
```

Both problems are related because certain FTP utilities require an explicit 'newline' character (CR or ASCII 13) to terminate each FTP command. It appears that individual commands sent without the 'newline' character are wrapped into a single string thereby confusing the FTP server.

"We do use an ftp wrapper that may be causing the login command to fail on the ftp server. However you can get in by using the following commands which I use in a perl script to automate uploads to the server."

```
/usr/ucb/ftp -n ftp.esri.com
user anonymous klewis@esri.com\n
bin\n
put up.tar\n
quit\n
```

Kent Lewis
Systems administrator
ESRI-Redlands

The above code can not be implemented in the FTP upload program because using explicit newline characters make it incompatible with the vast majority of FTP servers that are accessed by the GMS conversion utilities. Both problems demonstrate the considerable amount of variation between FTP servers that includes differences in the settings of individual server implementations as well as the platforms on which they operate.

The login and password text fields of Figure 5.1 are designed to provide the user

with a maximum degree of flexibility in accessing any on-line FTP server. The user's login name and password can be entered into these fields to access data files from their own directories as well as from anonymous FTP servers. Some FTP servers do not provide anonymous login so the current facility was extended to incorporate login from password protected servers.

When testing the conversion utilities, it was apparent that a considerable amount of time was taken in the upload and conversion of moderate to large data files. The factors that influence file upload include file size, server load and network bandwidth. These factors are purely hardware related and could not be directly addressed during the development of these conversion utilities.

The relatively lengthy conversion times associated with the conversion utilities (particularly in the case of the TIGERARC utility) is a definite impediment to their widespread use over the Internet. For example, by uploading a TIGER/Line zipped file of 2.5 megabytes in size from a local server, the process of conversion to an Arc/Info coverage using the TIGERARC conversion command can take between 35 and 45 minutes. Such response times are the primary reason to allow users to view on their browsers the execution and output of the AML programs associated with all the utilities discusses in this section. Therefore, during file conversion or coverage plotting, the user is able to view each AML command as it is being executed thereby providing continuous feedback during the entire operation of the program. Again, the system is not very user friendly with user trials confirming that the excessive length of time to conduct some of the conversion operations gives the impression of a system that 'hangs' thereby decreasing the reliability of the software.

The problem is not with the scripts that constitute the on-line conversion utilities, but with the conversion software itself. The performance of the conversion software used for the GMS conversion utilities on excessively large geospatial data sets is far too poor for the relatively short response times demanded by Web users. The proper design of feedback mechanisms that provide a time-remaining indicator or track all of the stages of converting one format to another is a much needed feature for this type of software. The GMS conversion utilities are equipped with a caching mechanism, which stores previously converted data sets in a reserved directory. This feature prevents the continual conversion of all types of geospatial data sets by

serving previously converted data to the user thereby saving time and unnecessary processing. The issues surrounding the processing of very large geospatial data sets by GIS software (including conversion software) are indicative of the complex nature of GIS data structures with typically many variable length data records spread across numerous linked files (Healey et al., 1998). In addition to GIS operations being compute and I/O intensive, they may also be composed of several applications containing sequential components. The performance impediments of GIS software due to the structure of GIS data and the scarcity of network bandwidth on the Web severely affects the ability to deliver this functionality to a vast number of geospatial data-users.

Examination of the file sizes and contents before and after upload suggests differences that can be attributed to the way in which local/remote servers and platforms format data within a file. For instance, uploading DLG data using the GMS FTP client slightly alters the data upon upload rendering it useless for data-conversion with the DLFARC command. Also, uploading TIGER/Line data in zipped format could not be uncompressed with the zip utility; thereby, preventing it from being used within the TIGERARC conversion utility. Among the probable causes of such a problem is in the way different computing platforms handle data that is represented as multiples of bytes.

This is known as the *endian* problem, which specifies two distinct representations of binary data among operating systems. Since a byte is 8 bits, a normal integer value is stored in 32 bits or 4 bytes. Little endian systems store the least significant byte in the lowest addressed byte while big endian systems store the most significant byte in the lowest addressed byte. For example, consider the integer value 66051. The value is represented in hexadecimal with 2 digits for each byte, hence 00010203. Assuming this is stored in a 32-bit integer at address 400, the bytes would be organised as follows:

Byte	Little-endian	Big-endian
400	3	0
401	2	1
402	1	2
403	0	3

Some little endian systems include Intel 80x86, Pentium, Digital VAX, Alpha, and

Decstations. Big endian systems include Motorola 68000 family (used in Macs and Sun 3), Sun SPARC and HP. The same issues apply to 16, 64 and 128 bit integers and to floating-point values. Therefore, depending on whether a system is big-endian or little-endian based, binary data can not be used between the two types of systems prior to conversion. Using the Internet to exchange binary data only accentuates this problem because a file's origin is often lost due to the repeated transfers from system to system.

Accessing remote FTP servers revealed certain problems with the on-line DXF data-conversion utility. Users attempting to convert E00 files with this utility discovered that either the DXF conversion command failed to convert the E00 file or the GMS FTP client could not upload the E00 file to the local system. The former scenario is described in the email message below:

"I tried using the FTP utility. The URL is FTP://ftp.gbutler.com/ files c131plac.e00, c131quad.e00, and c131rds.e00. I also tried downloading one of your sample files and got the same message. The program says the files are uploaded then it gives the message that it is "Importing interchange file and converting coverage to a dxf file", then the message "The DXF file can now be downloaded". Down below the Download file button there is the message "A DCL Error Occurred". If I activate the download button I get the message "Data-conversion was unsuccessful"."

Mark McReynolds

The above example shows the successful retrieval of the E00 file from the remote server but the conversion utility failed to successfully convert the file to the desired DXF format. The later scenario is described below by a user of this conversion utility who also includes the feedback from his FTP server outlining the step-by-step process of the file transfer.

"We have been trying converting compressed E00 files via your server (actually my client). Everything seems to work, only during transmitting we get the following message:

220 bb50 Microsoft FTP Service (Version 3.0).
Connected to 134.58.57.50.
331 Anonymous access allowed, send identity (e-mail name) as password.
230 Anonymous user logged in.
200 Type set to A.
200 PORT command successful.
150 Opening ASCII mode data connection for jim/11001net.e00(433135 bytes).
426 Connection closed; transfer aborted. local: U15:[ANP.WWW.SCRATCH]20E03C05_5353.E00;1

remote:jim/11001net.e00
221

As you can see the transfer is aborted. Can you please help us out?"

James Stada
Univ. Leuven, Dept. Burgerlijke
Bouwkunde, Belgium

Notice the verification information was properly sent and accepted by the remote server. The file transfer mode was also successfully defined. The server feedback shows that the origin of the file was from a Windows-based system and its transfer was eventually aborted closing the FTP connection. More detailed information on the DEC operating system used to implement GMS can be found in DEC C, 1994 and DEC C, 1996.

Operating systems use different methods to encode data within a file particularly with regard to line delimiters. For example, the standard format for UNIX systems uses a single line-feed as a line delimiter. In contrast, DOS, Windows 3.x, 95 and NT use a carriage-return and line-feed pair in that order. The Mac operating system uses a single carriage-return character to delimit the end of line within its files. Therefore, a Mac HTTP server must add line-feeds to text documents so that they conform to the Web clients of other OS platforms (Yeager and McGrath, 1996). Files in E00 format must be transferred over in ASCII mode. If E00 files are corrupted during transfer, then they can not be used with Arc/Info data-conversion commands. This makes these types of files vulnerable to the differences in how they are represented by various computing platforms. Transferring E00 files in binary format using FTP will not work in any of the conversion utilities.

In both of the examples above, the solution might be as simple as using an additional format converter to ensure compatibility before file transfer between computing platforms. The CONVGIF utility (Dowers, 1995) is one such program, which converts files to VMS compatible format of stream-linefeed when transferred over from DOS/Windows systems with a record-format of stream-linefeed and carriage-returns. The program changes the record-format to fixed 512 byte records. However, CONVGIF can not be used on E00 files because it alters the format of the records, which prevents the files from being used within the Arc/Info conversion commands. Many servers also convert the file as it is being transferred to assure

compatibility with the client's operating system. This means that servers must know the record formats of the clients, which is not always possible. VMS, for example, has a rich set of file formats with each file having the file format contained as part of the file header and not just inferred from the file name. This doesn't happen on DOS, Windows or Unix systems where a file is defined as a stream of bytes and their interpretation is left to the application. The lack of interoperability when transferring geospatial data files over the Internet requires automated FTP clients (as used in the GMS conversion utilities) to have prior knowledge of the computing platform or server from which the data file originates. The subtle differences in data formats between computing platforms directly affects the operation of file retrieval and on-line GIS software.

5.4 Integrated WWW search-engine

The search-engine developed for GMS is designed to integrate the metadata stored in the metadatabase, the geospatial data located at various Web sites and the conversion utilities developed as part of the GMS implementation. The GMS search-engine draws metadata from the database and sends it to the Web browser as HTML executable buttons. When the user queries the database, the retrieved metadata is parsed of any relevant information that links it to the GMS conversion utilities, the Web robot or any on-line graphic images. The relevant metadata information is placed within HTML executable buttons, which (depending on the information contained) either converts geospatial data from one format to another, releases the Web robot (Geobot) to retrieve information from a remote Web site or displays a graphic image on the browser.

At present, such a feature is unique among metadata query and retrieval software on the Web. Because metadata from the database is fed directly to the on-line utilities via HTML executable buttons, the GMS search-engine is very dependent on how the metadata is originally entered into the database. For example, if the format acronyms for each geospatial data set are not properly specified in the *Format_Name* column or the URL addresses contained in the database column *Network_Address* are not of a

standardised format, then the HTML executable buttons constructed by the search-engine can not operate. The same is true for the information contained in the *Browse_Graphic* database table; without a proper file name or network address for the graphic file, the executable button shown in Figure 4.13 (Chapter 4) can not download the graphic file from a remote Web site.

Another feature lacking in the present GMS search-engine is its inability to access more than a single metadata server. The search-engine is designed to only access and search the GMS metadatabase. However, using the TCP/IP (Terminal Control Protocol/Internet Protocol) library supplied with the C/C++ compilers of most major operating systems, it is possible to construct code that sends the database queries to multiple on-line metadata servers. This is also possible in the Java programming language by using its networking libraries. For multiple server access, it is important that the database schema is the same for all the on-line metadata servers. If the design of the search-engine is validated through a proper user-interface, query construction and database output, then the networking code can be added allowing the software to access other servers.

Among the main critiques of users accessing the GMS search-engine is the small size of the image-map used for the circular-area search routine. The rectangular and quadrilateral area-search routines (see section 4.1 in the previous chapter) could not be effectively used due to the difficulty in entering geographic co-ordinates within an HTML form. Most users indicated the need to develop some sort of mouse driven graphical interface that allows one to drag out circular, rectangular and quadrilateral geometric shapes for all area-search routines. Such enhanced functionality can be developed using a Java applet or a browser plug-in. The advantage of using plug-ins is that being fully compiled applications they can operate very efficiently as an integral part of the Web browser. However, such external applications must be downloaded off the Web for the proper platform and manually installed on the local system. A Java applet is automatically downloaded and executed by the local Web browser upon accessing a remote Web site. The disadvantage of using applets is they tend to be quite slow because the application must be compiled each time the user retrieves the code. The performance of the search-engine was rated as good by a majority of users with most queries taking between 10 and 20 seconds to complete.

The metadata is retrieved within a few seconds from the database, most of the remaining time is spent scrolling the typically large amount of information to the Web browser. Metadata records (see an example metadata record in appendix E) tend to contain a considerable amount of information.

Most applications operating on the Web must also deal with issues regarding concurrent access of system resource by multiple on-line users. This is particularly important when allocating system resource for a process such run-time memory or temporary storage space. GMS stores files with unique names based on process id and CPU time used. The VMS file system (Record Management Services) allows file sharing for READ and WRITE, so a file can be opened by one process while being read by another. This is rarely useful in an HTTP context, because usually any process writes in information unique to itself.

However, what about the values of CGI environment variables when two or more of the same processes are run simultaneously? These variables are assigned to DCL symbols for use within server-side scripts. Local symbols in DCL assigned with the '=' operator are not affected because the scope of the symbols do not extend beyond the process within which they are created. This is true for all platforms and scripting languages. On the other hand, global symbols assigned with the '==' operator are available to any client that follows afterwards when connecting to the same HTTP process. If two or more scripts run simultaneously then they are each identified with separate processes. Other scripts reuse the processes once the previous ones have terminated. As a result, global symbols are available to future script execution but the processes that connect to these scripts and the ones that follow can not be predicted.

Finally, the user-interface needs to comply with clients who enter date information beyond the year 2000. This modification only consists of adding more dates to the HTML <select> element. The Oracle DBMS handles dates beyond 2000 because its actual limits stretch from the year's -4712 to +4712 provided the default format of DD-MON-YY (date-month-2 digit year) is not used (ORACLE7™ Server, 1992a).

5.5 Geospatial Web robot

The Geospatial Web robot (Geobot) is among the first attempts to catalogue by automated means the URL addresses of metadata files, geospatial data sets, associated graphic files and other geographic information. The information is stored in a centralised database that can be searched much like other Web robots. Geobot successfully demonstrates the ability to automatically search and catalogue numerous WWW servers by traversing Web pages, examining the file extensions contained within URL addresses and identifying specific types of geographic information. The utility is only a first step in actively using software agents to organise information far more quickly than can be accomplished manually. Geobot is missing some key features common in other Web robots such as the ability to simultaneously access numerous Web pages and the adherence to the robot exclusion standard[1]. Robots should also be able to identify themselves, their creators and any additional information about the software that is of interest to administrators of Web servers. The speed the robot operates and the frequency it accesses any given host should be moderated either by the robot itself or by its creator. Also, Robots falling into infinite loops is a problem of considerable importance described in greater detail later in this section.

A number of problems associated with Geobot were discovered through testing and user feedback. Perhaps the utility's most obvious problem is its periodic propensity to parse the same HTML page twice thereby causing the robot to loop through the same page within a site domain. The size of these loops can vary making it difficult to detect the error. On occasion, the robot has been seen to traverse hundreds of pages before reverting back to an HTML page it parsed at the beginning of its operation. An error such as this prevents the robot from proceeding any further within a site domain. The error was eventually corrected through a mechanism that monitors the URL addresses entering the database and 'DO' file to ensure no duplicates occurs (see the flow diagram in Figure 4.39 of the Geobot parsing algorithm). Improvements to the parsing algorithm in locating URL addresses within

[1] An established mechanism for WWW servers that consists of methods used to exclude robots from parts of the server. It is implemented by creating a file on the server with a standard path (/robot.html) and contents detailing the nature of the access policy for robots visiting the site (Eichmann, 1994).

a document has also helped in this regard. This includes the addition of support for HTML frames and partial URLs as well as accounting for all the major protocols used in today's Web documents.

Another cause for concern is Geobot's rather slow database search-engine. Because this utility searches for user supplied keywords within each database record, the search time is usually long for relatively large databases. The Oracle DBMS provided optional structures such as indexes, clusters and hash clusters that can be associated with tables to increase the performance of data retrieval (ORACLE7™ Server, 1992b). For example, Web robots have tools such as categorised catalogues and inverted indexes where the Web pages are stored in database rows and the columns are designated with keywords (DeBra and Post, 1994; Randall and Randall, 1995). Other methods include latent semantic indexing where documents with similar word profiles (defined from inverted indexes) are clustered together. There are also grammatical parsing techniques and concept hierarchies that provide more intuitive ways for the software to interpret database queries.

There are a number of areas discovered during the development and testing of Geobot that enable the software to work faster and more efficiently. One of the more obvious ways of redesigning the robot is to integrate the crawling algorithm, and hence the Fetch_HTTP utility, directly into the parsing algorithm. Since both the parsing algorithm and the Fetch_HTTP utility are written in C, it makes far better sense to link the latter directly into the former code rather than having both the programs controlled by the CGI script (or crawling algorithm). Scripting languages like DCL tend to be a lot slower than compiled languages such as C. So combining the three robot operations, uploading, parsing and crawling into a single Pro*C program results in the code being far more efficient and portable.

Geobot users suggested a number of improvements including the development of certain mechanisms that allow the client greater control over the robot while it is in operation. At present, the user's only option is a button that stops the robot. The ability to repeatedly start and stop the robot as desired is a welcomed development. As a consequence, the user is able to direct the robot to search for specific file extensions (by editing the format.txt file) as well as certain pages within a site domain. The user is not able to specify the search of a group of pages or sub-

directories. However, it is useful to know that the database keeps track of Geobot's traverse pattern preventing multiple users from traversing a site domain previously visited by the robot.

Presently, the database maintenance utility developed for Geobot allows the user to remove broken or redundant links from the database. Future developments to the robot can consist of automatically maintaining its database using a small software agent that validates and updates all the links contained within the database. Ultimately, this softbot (Cheong, 1996) forms a basis for further improvements consisting of added functionality that incorporate other types of software development models such as Artificial Intelligence (AI) techniques. These include expert levels of functionality within well-defined domains to more easily control the robot while in operation as well as imprecise or 'fuzzy' sets or categories that allow the robot to understand more intuitive commands from users. Improved GUI functionality can make Geobot easier to use and allow it to display information in multiple dimensions, which turns the robot into a powerful and autonomous information gathering device (Etzioni and Weld, 1994).

5.6 Chapter conclusions

This chapter evaluates the issues, problems and procedures of each GMS component through their implementation in terms of system usability, performance and their relevance towards enhancing the access of geographic information over the Web. CSDGM metadata files tend to be difficult to parse and store within relational databases due to the lack of structural uniformity inherent within the standard. However, the GMS metadatabase and the on-line parsing and database entry utilities attempt to do just that while at the same time examining the problems raised as a result of implementation. Parsing programs are fallible and the resulting errors can be difficult to detect given the amount of data present in a typical metadata file. Storing the information and subsequently editing it over the Web requires administrators to maintain adequate referential integrity rules for the relational database as well as proper security and access controls. Ultimately, the metadata

stored in the metadatabase can be directly used by the GMS search-engine and geospatial data-conversion utilities. These on-line applications can query metadata, retrieve referenced data sets and convert them into a desired format using a data-switch-yard. However, differences in the formats of records from different operating systems posses serious barriers to interoperability for such Web applications. The limitations of the user-interface of the GMS search-engine particularly with regard to the absence of vivid GUI capabilities is also noted and examined in this chapter.

CHAPTER SIX

CONCLUSIONS AND FUTURE RESEARCH

A prototype system developed as a part of this research is designed to facilitate the understanding of fundamental issues and software concepts that influence how geospatial information is represented, catalogued, searched and processed over the Internet. The Geospatial Metadata Server (GMS) clearly demonstrates the convergence of several different technologies designed to operate in combination. These technologies include an HTTP server to send and retrieve geospatial data and meta-information from the Internet, a RDBMS to store and query metadata information and a GIS to process geospatial data. A number of utilities developed for GMS using these software packages are designed not only to operate independently over the Web but also to exchange metadata amongst them and form a cohesive system to catalogue, query and process geospatial data.

The system can be expanded to demonstrate the operation of each GMS component on multiple Web servers as distinct pieces of software that communicate through the exchange of metadata information. The GMS components are originally designed independently from other components of the system and each are implemented with their own user-interfaces that are accessible over the Web from any standard Web client. Only a single HTTP server is used during the development and evaluation of each GMS software components. In order for a system such as this to be of service to the largest possible number of on-line geospatial data-users, its components should be able to operate on multiple Web servers. There are limitations associated with network performance in managing rather involved database and GIS operations over the Internet (Laurini, 1995). However, software that is distributed throughout the Internet allows users from all locations of the network to browse metadata, access geospatial data and perform on-line data-conversion or GIS analysis. Alternatively, clients accessing a particular HTTP server can process their requests and relay them

to other remote servers that operate one or more relational databases or GIS applications providing a variety of different forms of data storage, processing and analysis functionality over the Web. In this manner, one can conclude that the storage, catalogue, query, and process of geographic information such as geospatial data and its associated metadata exists not only as a cohesive interconnected system, but also as a distributed system comprising of a collection of applications on multiple Web servers.

The implication for GIS as a whole is that the technology that drives all aspects of the Internet is increasingly being adapted and merged with existing GIS technology in order to provide geographic information, in all its various forms, to an increasingly larger community of users. The design of GMS demonstrates the ability of the software to operate seamlessly between different physically remote Web servers. It also shows the ability of the WWW to distribute fundamental spatial data cataloguing and analysis operations throughout a large communication network such as the Internet.

This is achieved by incorporating the most commonly used Internet standards and protocols with state-of-the-art RDBMS and GIS technology. For example, the FTP file upload utility developed for a number of GMS components uses the File Transfer Protocol (FTP) as a means to securely move files across different computers and computing platforms. FTP requires clients to provide verification information to a server, which lists folders and transferable files on its system directories (Naik, 1998). The protocol is able to download files in ASCII or binary formats enabling the construction of on-line geospatial data archives on servers where access is easily regulated by a username/password or anonymous remote login. Both HTTP and FTP use TCP/IP (Transmission Control Protocol/Internet Protocol). But the latter must establish a TCP connection for transferring data and commands using an FTP user/server model configuration. Alternatively, the Hypertext Transfer Protocol (used to construct the HTTP file upload utility for a number of GMS components) is a stateless and connectionless transfer process that understands the URL standard for addressing network resources (see section 2.5.2 of chapter 2). This fast I/O protocol

consisting of a small number of commands encapsulates requested data in MIME[1] format for transfer between a client and server. MIME makes HTTP extensible for multiple media formats but the primary advantage of using it over FTP is in its request/response model that allows information to traffic between clients and servers using TCP/IP. The implication is that geospatial data sets of a range of file sizes can be quickly and conveniently transferred over using a simple addressing mechanism.

The HTTP and FTP file upload utilities are an integral part of GMS. Each component of this on-line software system explores a variety of technical and institutional issues relating to the operation of future metadata catalogues and GIS applications over the Web. This research concludes by identifying how these issues are addressed and their implications towards the adoption of the Internet as a new computing architecture for GIS.

6.1 An on-line federated database for GIS data

Given the complexity of the CSDGM standard, the GMS database demonstrates that a sizeable portion of this standard can be implemented within a relational database. The database for GMS is implemented within an RDBMS using a database schema incorporating some of the section and element names found in the CSDGM metadata standard. The database shows its versatility in providing a relatively detailed foundation for metadata storage and access upon which to build a number of other applications designed to operate over the Web. As a result, the use of relational databases that are accessible over the Web can be an important tool to store and query large quantities of standardised metadata. The more standardised metadata that is placed on the Web, the easier it is for users to quickly and accurately find geospatial data and other forms of geographic information among the multitude of HTML documents, FTP directories and HTTP servers.

The Oracle8i™ Spatial software package now provides an integrated set of functions and procedures that enable spatial data to be stored, accessed and analysed

[1] Another protocol primarily used to transmit e-mail messages but it also specifies how characters are encoded by transforming data from an 8-bit format to 7-bit ASCII characters, which can be successfully transferred over the Internet (Naik, 1998).

in an Oracle database. Web site administrators can add useful spatial queries to the on-line applications that access a centralised or networked database containing geospatial data. It also allows data providers to store, retrieve and manage very large geospatial databases containing vast quantities of GIS data (Oracle, 1999a; 1999b; 1999c). Oracle8i™ Spatial also has a Java based API for managing, accessing and manipulating Oracle Spatial object types when provided with a class hierarchy for Geometry and respective subclasses. Pollock and McLaughlin (1990) review the need to use object-oriented DBMS technology as an approach to unify the management of spatial data. More traditional database management systems find it difficult to clearly model and efficiently manage complex spatial data object as handled by GIS software. As a result, traditional GIS software is constructed in a hybrid fashion where lexical and geometric component of spatial data are managed separately.

However, even with Oracle8i™ Spatial, site administrators who manage on-line metadata and geospatial data dissemination sites must still implement on-line storage mechanisms for standardised metadata (i.e. CSDGM) based on database technology. Oracle8i™ Spatial can provide a useful tool either to GIS vendors who need to link their software applications to a relational database used throughout the industry or to GIS users who require powerful tools for spatial queries. However, the core of any on-line metadata and/or geospatial data dissemination site is the metadata database, which must be modelled after a recognised national or international standard. The database must be constructed regardless of using relational or object-oriented technology. One key advantage of using object-oriented over relational technology to design the metadata database is that the hierarchical nature of many metadata standards (as in the case of CSDGM) makes it a more natural modelling environment. This research demonstrates that present metadata standards can be partially or fully implemented within heterogeneous database servers that are accessible by clients over distributed networks comprising, among other things, the Internet.

The implication is that federated metadata databases can be constructed and implemented far more easily using current Internet technology than the multiple-vendor database server so often associated with custom built local area networks

(LANs). Large portions of client/server applications consist of database servers. These servers consist of client applications requesting data and data-related services from a database server or SQL engine, which can respond to the client's requests by providing secured access to shared data (Orfali et al., 1996a). Ideally, in a single-vendor database server environment, clients should be able to send SQL commands to one or more database servers over a network such as a LAN and have data subsequently returned from multiple sources. In reality, heterogeneous database servers that are manufactured by multiple vendors are used by organisations throughout the IT industry. Because of extensions placed within SQL by competing database vendors, the language never succeeded in becoming a common API for the entire database industry. This made common application development across multiple-vendor database systems all but impossible. As a result, the seamless and interoperable exchange of information between clients and database servers from multiple-vendors is currently only possible over distributed networks as a system of *federated* databases, which are loosely-coupled using middleware that rationalises the differences between vendor-specific database systems systems (Abel et al., 1994).

The advent of the WWW and the standards and protocols on which it is based, provides a far easier medium to construct federated database. With the introduction of some key Web technologies in 1995, the Web quickly morphed from a medium for transmitting static Web documents to one that is an interactive client/server environment. In such an environment clients such as Web browsers are able to send requests to Web servers. The Web servers operate back-end systems such as database servers through custom-built software applications. Information is retrieved from one or more database servers and sent back through the Web server to the client as a response. The entire system is known as *3-tier client/server architecture* as envisioned using the Web. The architecture identified as the first tier, is the Web client or browser, connected to the second tier, the Web server, via middleware consisting of HTTP and FTP on top of the TCP/IP protocol. The third tier consists of the back-end systems including database servers, GIS packages, operating systems, etc.

It is this, 3-tier client/server architecture of today's Web that provides the means to access vast quantities of metadata and geospatial data from a collection of federated database servers to be used in a variety of analytical software such as GIS and other related Decision Support Systems (DSS). The primary technological developments that allow existing database systems to combine their facilities over the Web include standardised Web browsers, which support fill-in forms, tables and a protocol known as the Common Gateway Interface (CGI). HTML *forms* are introduced in the HTML 2.0 specification published by the WWW Consortium (W3C). Forms allow users to embed text-entry/text-area fields, radio boxes, selection lists, check boxes and submit buttons within HTML documents, which collect information and send it to a CGI server application whose location is specified with a URL address within the form definition. The Web server receives the information from the client and sends it to the CGI application, which in turn operates any back-end programs using the supplied parameters. In the case of a back-end program such as a metadata database server, the results of a database query can be sent back to the client Web browser and placed within a tabular format using HTML *tables* (as specified by the HTML 3.0 standard). The technologies just mentioned along with Web clients, Web servers and the standards/protocols comprising the middleware in between form an effective means of extending the 3-tier client/server architecture to a *federated database system*. Such databases are capable of disseminating metadata and geospatial data from a vast number of physically remote database servers.

GMS can be expanded to implement the database schema on numerous database servers that are accessible over the Internet via Web servers operating CGI server applications. The current implementation of the GMS metadatabase uses a search-engine located on the same server as the database schema, which performs database queries and reports the results back to the user's Web browser. A situation where the search-engine is capable of querying multiple Web servers each containing a database patterned on the one used for the GMS metadatabase is a poignant demonstration of a federated metadata database that adheres to a full implementation of a recognised national or international metadata standard. See Figure 6.1 for a representation of such a system.

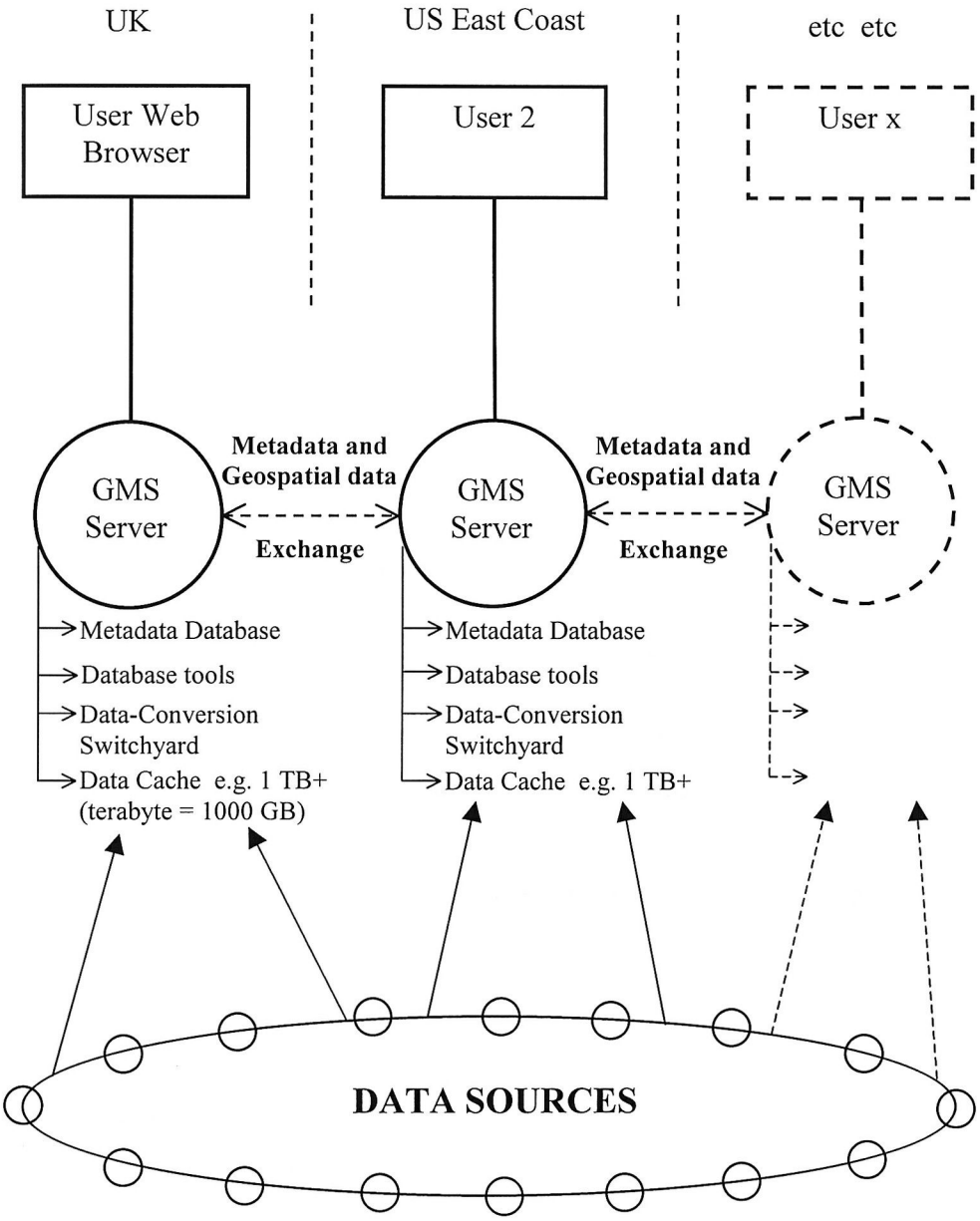


Figure 6.1 A diagram of the proposed federated metadata database where users are able to query multiple servers containing standardised metadata and access geospatial data referenced through hyperlinks for conversion to one or more different data formats.

The system demonstrates how clients can access regional GMS-type servers (or other servers depending on server-load and bandwidth) that implements metadata databases containing standardised metadata. The metadata contains hyperlink

references to geospatial data sets located throughout the Internet. There is a data-cache to improve overall system performance by storing frequently used data files (see below) locally. These large on-line data cache can exist either as massive directories of geospatial data files or within back-end Web applications such as database management systems (Stonebraker, 1991; Stonebraker and Dozier, 1991; Guptill, 1992; Stonebraker, 1993). There are additional tools to exchange or replicate metadata between different GMS servers, convert geospatial data from one format to another (see section 6.3) and tools to enter/maintain metadata within each database.

It is possible to design and build tools to entry/modify metadata on-line which operate on a single server or in a federated fashion across many diverse database servers. Database tools capable of operating seamlessly over the Web are vital in maintaining large and distributed information resources. This includes ensuring on-line metadata databases contain the most up-to-date information and that any outdated or redundant metadata is either modified or deleted. The metadata parsing and database loading utilities (MPARSER and MLOADER) developed for GMS are examples of such tools. They operate on the local metadata database by providing an effective demonstration whereby entire metadata files can be catalogued by quickly parsing and entering them into an on-line relational database. The first utility parses a metadata file and the second utility enters the values of their elements into the appropriate tables and columns of a database record.

The metadata parsing and database entry programs are developed to address the excessive size of metadata files and the difficulty associated with manually entering them into on-line metadata catalogues. The ability to extract information from relatively large metadata files and enter it into a database can reduce the amount of time spent by site-administrators in maintaining the currency of their on-line databases. Also, such a mechanism increases the amount of metadata that can be quickly and accurately placed onto the Web. Systems such as GMS use these database maintenance tools to encourage providers of metadata to store such information within centralised database servers. These database servers can be networked with other servers, which are searchable using Web browsers thereby

facilitating the on-line access to geospatial data and related geographic information to a potentially vast community of users.

Another application developed alongside the GMS metadata database is the on-line database entry and modification utility. The utility demonstrates that using an on-line relational database to store metadata allows geospatial data-providers a very high degree of control over their metadata records held within a centralised metadata-cataloguing site. The utility is a series of HTML forms designed to operate on the GMS database by allowing data-providers and site-administrators to manually enter, modify or remove specific metadata records from an on-line metadata cataloguing system. Each metadata record is identified with a username and password. Its owner is able create new records, edit and delete existing records using SQL (select, insert, update and delete) operations on any table or column of that particular record. As a consequence, the system substantially reduces the amount of database maintenance required by the site-administrator because each data-provider, wherever they are located on the Web, is responsible for maintaining their own metadata records.

A tool that gives metadata creators the ability to maintain their own metadata records is a substantial step towards providing centralised and accessible metadata storage facilities on the Web. On-line federated databases containing very large numbers of metadata records are difficult to maintain in terms of inserting new records into the database, updating current records and removing any dated or irrelevant information. This would require a substantial number of dedicated personnel to perform what is considered largely mundane duties in maintaining each database on the network. The GMS on-line metadata editing utility shifts the most routine tasks associated with maintaining individual database records to those who create the actual content of the database. The utility requires only a subset of a metadata file to be entered at any one time thereby dividing the task into manageable parts. This encourages people or organisations to more easily construct metadata records for their geospatial data sets and place them within centralised and searchable on-line databases.

In a general sense, the GMS utilities discussed in this section demonstrate the ability to extend the functionality of back-end systems supplying geographic information (i.e. database server) to the Internet. The 3-tier client/server architecture

for the Web is an effective applications development platform for distributing this type of information. Using CGI, organisations that provide on-line geographic information can implement relatively inexpensive and easily available technology. Such software comprises of cheap, user-friendly, multiple-platform Web browsers that request information from centrally located Web servers designed to act as gateways for gathering and distributing metadata and geospatial data. However, CGI operates in real time and suffers from performance problems or server-side bottlenecks due to the long and convoluted process of sending requests from the Web client to back-end applications. CGI is a stateless protocol meaning once a response from the server is sent back to the client the server proceeds to the next request forgetting all other previous transactions. HTML *hidden* fields such as the ones used in various GMS components are aimed at circumventing this problem but it is not an ideal situation.

Extending functionality to back-end systems, particularly database servers can also be achieved using the comprehensive set of tools provided by the Java language for accessing databases over the Web. The Java Database Connection (JDBC) standard consists of a set of Java classes that can interface with all the major industrial SQL databases over the Web through common Java applications. This technology provides a single call-level SQL-API for the Java language that allows developers to write code that works with all JDBC-supported database products (Jepson, 1997). The similarities between CGI and JDBC in accessing back-end database applications are that they both use the same protocol, TCP/IP, to transfer information between clients and server. However, CGI servers receive their parameters from a Web form via HTTP. JDBC servers use their own URLs that are structured as a JDBC protocol indicator (jdbc:), a subprotocol indicator specifying a database driver (e.g. odbc), a hostname, a port and a database name. For example,

jdbc:odbc://some.host.com:1112/a_database

Both CGI and JDBC can be used to construct search-engines designed to query geographic information from multiple on-line federated databases and both hide the vendor specific peculiarities associated with the APIs of different database servers. However, JDBC allows the construction of Java applets that are independent of any particular database making applications portable to a wide number of database

servers. The object-oriented nature of Java allows developers to construct components using JDBC that can be incorporated into database applications. These components can also be reusable allowing prototype systems to be built more rapidly than conventional methods.

Therefore, large systems such as GMS can be assembled more easily using an already established 'toolkit' comprising of reusable components. Because these components are portable, they can be easily implemented on other Web sites operating a JDBC-supported database. Alternatively, CGI scripts must be ported to different operating system running different database servers. The use of JDBC represents a further evolution in the construction of a federated network of metadata database servers outlined in Figure 6.1. The principle is essentially the same; the only thing that differs is the technology used in its implementation. Developers can use a standard API to construct database queries and subsequently send them to remote database servers that are equipped with the proper JDBC supported drivers.

6.2 An on-line Decision Support System for GIS data

The development and implementation of Decision-Support Systems to analyse geographic data and assemble it into a variety of different report formats is an important area of GIS research. Today's GIS packages incorporate many features associated with DSS, such as the ability to construct sophisticated queries that answer "what if" type questions. Patterns within geospatial data can be recognised and compared with other data sets from different time periods or locations. Results can be viewed through a variety of intuitive means or sent to other applications such as spreadsheets, statistical or related applications. The entire process is a continual evolution in GIS technology by making these systems easier in answering questions about the state of our physical surroundings from a geographic perspective and to place this technology in the hands of a wider number of users.

DSS tools within the client/server industry are continually evolving becoming increasingly more efficient and powerful in their ability to extract useful information from raw data. Such software tools are becoming increasingly independent in their operation by requiring smaller amounts of interaction from the user. The simplest

class of DSS tools are those that query and analyse data. Delivering GIS technology over the WWW is an initial step in providing query and analysis functionality of geographic information to an increasingly larger number of users. As with the case of standalone GIS packages, the success of GIS functionality over the Web depends on how well these on-line tools can be designed for decision-support purposes. This includes their ability to access federated metadata databases, analyse on-line geospatial data, formulate sophisticated queries, report results using powerful multidimensional graphic techniques and operate independently towards overall data discovery (Moore et al., 1999).

The GMS conversion utilities are an attempt at demonstrating the operation of complex GIS operations such as data-conversion over the Web using geographic information obtained from a variety of on-line sources. Using the Arc/Info[2] data-conversion commands, GMS is implemented with three geospatial data-conversion utilities designed to operate on the Web that are capable of converting files in DXF, DLG and TIGER/Line to and from E00 formats. The three conversion utilities upload a geospatial data set from any remote Web site and return the converted file to the user via a hypertext link. Because E00 is used as a neutral data format, files can be inter-converted between DXF, DLG and TIGER/Line formats in what is known as a data switch-yard. The ease by which geospatial data can be converted from one format to another using the data-switchyard also demonstrates an effective means of using current Web technology to address the problems of interoperability between GIS software packages that use proprietary data and interchange formats. Since on-line geospatial data sets are referenced using URL addresses, the conversion utilities are capable of accessing such files located from any Web server. The result is a clear demonstration that such GIS functionality is capable of accessing data beyond the confines of the local server from the Internet where storage is virtually limitless and maintenance of geospatial data holdings is distributed across many Web servers.

GIS commands operating from many different Web servers can be pooled together to form a comprehensive on-line DSS which searches on-line metadata catalogues,

[2] The use of the Arc/Info GIS demonstrates the system in principle. A real production system could use either customised or better specification conversion tools.

retrieves geospatial data for analysis, collaborates through the exchange of information and collectively reports results. The conversion utilities implemented for GMS consist of a small portion of commands demonstrating the feasibility of constructing a wider collection of GIS commands for operation over the Internet. Indeed, some of the conversion utilities provide optional coverage processing such as building topology or changing the projection of a data set. Using the GMS file upload algorithm, it is possible to implement a wide range GIS commands for use on geospatial data located anywhere on the Web. The development of a data cache for the GMS conversion utilities (see appendix F for its implementation) shows that the use of simple caching techniques can substantially increase the performance of on-line GIS operations. In this manner, storing frequently used geospatial data sets locally can spare the system from having to upload the file(s) from the Internet.

GIS commands, particularly those used for data processing are relatively easy to construct and operate from a client's Web browser using the already validated file uploading algorithm that retrieves files using both HTTP and FTP protocols. More importantly, the on-line GIS commands are capable of operating from any Web server containing the appropriate software, enabling the processing of GIS operations to be distributed throughout the network. Such a system demonstrates the ability to use the Internet as a means to supply GIS functionality to the geospatial data-user community in ways that are not achievable through more traditional methods of software distribution.

Search-engines are used to query geographic information from one or more on-line federated metadata databases and geospatial data storage sites. The GMS conversion commands are examples of the data processing and analysis operations that can be performed on on-line geospatial data. However, these operations are not complete unless complemented with browse and query software that enables users to locate specific geographic information within a federated network of servers. One of the fundamental operations of a DSS is the ability formulate queries without requiring the user to develop additional software applications composed of complex programming code or macro languages. Such tools usually allow database queries to be constructed from fill-in-forms and the results are displayed in tables, bar/pie charts, histograms, line graphs or as geographic maps. All of these display

mechanisms can be used to convey relatively complex geographic information in a simplified manner over the Web. Many of these display mechanisms are currently in use with other geospatial data dissemination sites. However, in addition to simply querying and displaying geographic information over the Web, a powerful on-line DSS should be able to suggest possible methods of data analysis given the contents of the information queried by the search-engine.

The Web search-engine designed to operate with GMS demonstrates that command line arguments for on-line GIS software, such as the conversion utilities, can be supplied automatically from a metadata database. The GMS search-engine is capable of integrating the conversion utilities and the Web robot with the database. It can query database records and write the information (i.e. URL addresses and file names) to HTML form buttons, which are used to execute the on-line utilities using the information contained therein. The GMS search-engine provides the means to control one or more on-line metadata catalogues, geospatial data storage sites and GIS applications. As a result, Web applications can exist separately on different physically remote servers and each is capable of being accessed from standard Web browsers. The system components that comprise such Web applications can share metadata information and use it according to their particular functionality. The implication is that Web search-engines can be used as the primary instrument to integrate physically remote components of Web applications designed to describe, search and process geographic information.

The most powerful types of DSS tools are search algorithms known as Personal Information Agents (PIA) whose primary purpose is delegated to them by the user for the purposes of data discovery. Given the numerous types of software agents (see Cheong, 1996), the category that best places PIAs include those software entities that are delegated to perform a variety of tasks for the humans that use them. This broad category includes Internet Softbots, which are Web robots capable of exploring the hyper-linked documents for information and catalogue the results for future reference. For example, Geobot is a Web robot constructed for GMS that demonstrates the use of software agents in automatically finding and cataloguing very large Web sites containing geospatial data sets and their associated metadata. A PIA is usually more advanced than a Web robot capable of handling either specific

queries or ones based on fuzzy logic. The boundaries between the sets or categories of fuzzy queries are not precisely delineated, as is the case with more traditional query mechanisms such as SQL (Berson and Smith, 1997). PIAs are typically rule-based expert systems capable of searching for patterns in data that are designed to discover unsuspected occurrences within a federated environment (Riechen, 1994).

Geobot is a relatively simple software agent that performs a single query that starts the Web robot on a specified site domain in order to look for files of a particular file type. The software scans the HTML pages of a site domain for hypertext links to geospatial data files by examining their file extensions and supplies these links to users in the form of an HTML document or a searchable database. Due to a variety of factors, users and providers of geographic information are increasingly adopting the Internet as a preferred means of information exchange. Because of the sheer size of this medium and its exponential expansion in terms of the number of on-line servers and users, the use of automated Web software to search and index geographic information is becoming increasingly significant.

The present section describes some of the ways relatively established client/server and decision support technologies are making rapid inroads into the area of GIS. How could GMS evolve into a geospatial metadata decision support system or GMDSS? As shown in Figure 6.1, this research as already assembled the initial components of a GMDSS. Such a system would need a centralised or federated database to store metadata on the identification, origin, content, quality and location of on-line geographic information. It would also need a variety of database tools that allow site administrators and data providers throughout the Internet to easily update and maintain the database. On-line geospatial data conversion utilities would need to be implemented to convert geospatial data either between GIS data formats or to a common data format for use within the on-line GMDSS software. A data cache consisting of reserved disk space on one or more servers can be set aside to locally store potentially large geospatial data sets. This allows on-line digital display and analysis tools to access data without incurring any constraints due to network congestion when downloading such data off the Web.

For GMS to evolve into a decision support system, it must be augmented with sophisticated database search and query tools as well as utilities associated with

analysis and display of geographic information. The GMS search engine can be expanded to query the database on many more CSDGM compliant metadata elements (this is elaborated on later in the chapter). New search algorithms can be designed based on Java technology that allow users to search for metadata records based on location by panning and zooming through multiple map layers as well as selecting areas by drawing circular, rectangular and quadrilateral shapes. Java applets, browser plug-ins or specialised Web clients can be constructed to display the geographic and temporal positions of spatial (i.e. raster and vector data models) and non-spatial map attributes as cartographic models. Finally, a GMDSS would need on-line integrated analysis of spatial and attribute data such as data retrieval, classification and measurement functions, overlay and neighbourhood operations, connectivity functions as well as data transformations and editing functions. The reader is referred to Aronoff, 1991 or Laurini and Thompson, 1992 for a more exhaustive discussion on GIS theory.

The trend towards developing on-line GIS search, query, analysis and display functions is due to the phenomenal growth in popularity of the Internet/WWW and associated networking technologies. This research succeeds in identifying and addressing some of fundamental issues and problems associated with the on-line access of metadata and conversion of geospatial data. Consequently, it contributes towards the further development of on-line GIS. In time, such systems will be the primary means by which people store, access and use geographic information of the Earth.

The Internet is continually evolving (some may say, reinventing itself every five years) through the introduction of new technologies that address some of the security, performance, statelessness and visualisation problems so often associated with the Web. The next section provides a brief introduction to these new technologies and their probable future impact on the on-line access of geographic information.

6.3 The future of GIS data on the Web

The future direction of the work contained in this thesis is dictated primary by the needs of geospatial data-users and the limitations placed upon them using present Web technology. Since the completion of GMS, a number of technical as well as organisational advancements are in the process of changing the underlying structure of the Web by extending its usefulness while increasing its security. As was once the case with the Web itself, the new technologies now available to Web-based GIS developers are in response to the needs of a much larger IT industry. Some of these new technologies are relevant; others are not. It is the purpose of this section to review the benefits of these new technologies towards the further development of GMS. This includes the purposed role of GMS as a federated system of metadata storage and GIS processing software that not only supports the distribution of geospatial data but also its potential commercialisation.

The further development of both client and server parts of GMS can be advanced with some of the new and evolving Internet technologies that are now available. For example, Java applets, JavaScript, Cascading Style Sheets (CSS), Extensible Mark-up Language (XML) and compound documents (e.g. DOM, OLE and OpenDoc) promise to vastly improved the design of HTML documents and enhance their interactivity when displayed on client Web browsers. On the server side, a number of technologies are coming together based around two distributed object and component infrastructures. The following commentary focuses on these emerging technologies and their potential importance towards accessing geographic information over the Internet.

The Java programming language provides Web designers the tools to add interactive and conditional behaviour to Web pages and more importantly, cross-platform graphics, networking and multithreading capabilities (Flanagan, 1997 and Anuff, 1996). Java applets are used in a number of on-line geospatial data dissemination sites that provide fill-in forms, pan and zoom through map layers and animated graphics. Multiple threads or processes improve the interactive performance of graphical applications for the client by allowing tasks to be handled

simultaneously. Future developments using this programming language should concentrate on extending the functionality of client-side mapping and querying applications for geographic information. The language does suffer from performance problems because its code must be interpreted each time it is loaded into the browser. Also, competing implementations by different software companies have impeded the portability of its code across platforms. Future Web-based systems like GMS must take these factors into account before deciding to use this technology to enhance the client-side of their systems.

Like Java, JavaScript can also deliver executable content over the Web, but that is where the similarity ends. The two languages do complement each other in that JavaScript can control the behaviour and content of a Web browser but it does not support graphics or networking functions (Flanagan, 1997). Perhaps the most significant feature of JavaScript with respect to this thesis is its ability to supersede a number of functions associated with CGI scripts. For instance, it can generate dynamic and conditional HTML documents without the need to send information to the server, as is the case with CGI. However, JavaScript can not replace CGI. On-line applications such as the GMS parsing/database-entry or metadata editing utilities still need back-end applications like the ORACLE relational database to store, modify and query metadata. However, JavaScript can error-check or pre-process information by reading user input from form elements before it is submitted, thereby reducing the amount of processing performed by the server and increasing overall system performance.

As with HTML, JavaScript also supports Form objects (checkboxes, hidden fields, text fields, radio buttons etc.) whose values can be assigned to variables[3], which can be used in mathematical functions. Therefore, applications such as the GMS search-engine could convert the co-ordinates (retrieved by either of the location search-patterns) from degree to decimal values on the client. This in turn, which can reduce the amount of processing on the server. Web designers can also use JavaScript to load content in newly spawned browser windows as well as HTML frames. Such tools could allow users to view multiple on-line maps and/or map layers that are either downloaded manually or in response to a query from a search-engine. Also,

among the major problems of the Web is that HTTP is a stateless protocol. A number of GMS components use HTML *hidden* fields (see section 4.1) to preserve data returned from the server for use in subsequent requests by the client. Yet, this method of preserving state between HTTP requests is not elegant and it can pose a security risk by providing users access to certain behind-the-scenes system functions Yeager and McGrath, 1996). JavaScript programs can preserve state by reading and writing cookie[4] values. In this fashion, data of previous client requests are stored and transferred to the server. The server responds by generating HTML documents on-the-fly which may, in turn, 'set' additional cookies. The process repeats itself.

Another emerging Internet technology includes Cascading Style Sheets. CSS for HTML allows authors to apply typographic styles and spacing instructions for elements on a page, which are capable of superseding one another depending on their rules of precedence (Niederst, 1999). By specifying font sizes, line spacing, letter space, indentations, margins and element positioning for HTML documents, style sheets add much needed cartographic tools for representing geographic information through maps placed on the Web. Web sites are also easier to maintain using style sheets because they can be defined for multiple HTML documents so that any changes in fonts or line spacing translate throughout all the referenced Web pages on the site.

XML is not designed to replace HTML but rather extend its utility beyond the confines of simply providing content for Web browsers. The language gives more flexibility to developers by defining document structures as opposed to simply specifying how content is displayed on a Web browsers as is the case with HTML. Developers are now capable of creating custom mark-up languages defining specialised tags that reflect the semantic relationships of their Web content, which can not be defined using HTML alone (Mace et al., 1998). In short, XML extends Web content for use in a variety of other types of clients such as printer or plotters and specialised image processing and analysis packages. Therefore, people and organisations who use geospatial data are able to move beyond restrictive HTML

---

[3] JavaScript supports floating-point values that follow the standard 8-byte IEEE floating-point numeric format.

[4] Used by Netscape Navigator™ to store small amounts of data permanently or temporarily with the client, which can be used by Web sites to recall information about the client.

standards committees by defining and creating their own Web content as well as producing and distributing the WWW applications that utilise their content.

For example, XML is currently being used to define and standardise a vector graphic format through the W3C Scalable Vector Graphics working group. Therefore, software vendors are able to produce WWW applications around this newly defined vector graphic format for purposes of placing vector maps on-line. A major component of XML is its Document Type Definition (DTD), which allows developers to specify the logical structure or grammar of a type of document. The DTD defines the elements and their attributes as well as their relationships to a particular document type. The significance of this is that any particular document type, such as a CSDGM metadata file, can be defined and validated using XML parsing software. Private, government or academic organisations can not only transmit their own data over the Web but also provide the means to define the data. With XML an organisation can embed organisation-specific descriptors within the document along with the data itself. The organisation then simply provides parsing software to decipher the document and the data contained therein. Therefore, organisations are not restricted to a single document type, as is the case with HTML, but can define numerous types of documents tailored for a specific type of data.

Future Web browsers designed to parse XML defined documents could, in theory, handle many different types of document formats. This makes current metadata validating software redundant because of their ability to automatically verify metadata files. This is particularly relevant for metadata files described using the CSDGM standard because XML extensions allow metadata creators to specify additional functionality with respect to their data sets. For instance, the CSDGM standard can effectively describe a single data set at a time. However, the standard proves to be inadequate where there are multiple data set collections, on multiple servers and using multiple access points (HTTP, FTP WAIS etc.). In this case, XML allows the developer to define customised *tagsets* for distributed collections of GIS data. The extended CSDGM definition could include service, protocol, mirror-sites, alternate download sites, service administrator and much more. The reason for developing such a *tagset* is to make individual metadata documents more flexible

and portable, especially with reference to the CSDGM <Online Linkage> tag[5]. Given the GMS metadatabase implements this tag, placing a specific url:file inside causes the documents integrity to become very volatile because directories, filenames and file extensions can change over time.

The Document Object Model is developed by the WWW Consortium (W3C) for accessing a document's individual parts. DOM is one of the key components of DHTML, which, for the first time, introduces a platform independent programmatic interface to HTML documents. The proposed standard essentially adds object orientation to page layout design by treating HTML elements as objects that expose properties and methods within containers or documents. Developers are able to create dynamic styles and content as well as manipulate the positioning of attributes for the purposes of animation within HTML documents using any scripting language (i.e. JavaScript). Manipulating the components of an HTML document is a powerful concept that is redefining the term 'document' throughout the client/server industry.

Compound documents are visual containers for organising collections of components[6] through a framework of integration, which specifies the protocols used on the desktop as well as across organisations via the Internet and Intranets. These frameworks consist of established technologies such as Microsoft's Object Linking and Embedding (OLE) and Apple's OpenDoc document component architectures used to create compound documents and manage their respective components contained within. Compound documents allow users and software developers to group related objects, display them seamlessly in a window, store them in a shared files, and transfer them across networks to other desktops or servers. Imagine a compound document that contains components incorporating all the functionality present in today's operating systems, Web browsers, word processors, spread sheet packages, relational databases and of course GIS software.

The implications for using geographic information and representing it on the Web are enormous. The process of producing on-line maps is substantially easier because they exist as individual components that can be moved from place to place until

---

[5] See the *Citation_Information* table of Figure 3.1 in chapter 3.

[6] Components are interoperable objects or system-independent software entities that can be invoked across address spaces, networks, languages, operating systems and tools. Because components are objects, they display all the benefits of object orientation technology including the fact that they can only be manipulated through their interfaces (Orfali et al., 1996b).

assembled on a Web page using a simple scripting language. Imagine being able to cut-and-paste objects such as text, images, maps, tables and diagrams between different desktop applications as well as placing them directly onto Web pages. Icons representing geospatial data or metadata files can be dragged-and-dropped onto database icons, which stores and catalogues the information on a centralised remotely located Web server. Alternatively, icons representing the geographic information encapsulated within distributed objects can be dragged-and-dropped over icons representing local or remote GIS processing operations. In summary, GIS is no longer viewed as a separate application on of the desktop (importing and exporting geographic information in vendor specific file formats) but rather it is seamlessly integrated with other applications on the desktop as well as over the Internet.

The commentary above focuses primarily on technological advancements concerning the client side of using geographic information obtained from the Web. However, compound documents will incorporate all of the functionality of the Web clients that initially ushered in the WWW computing movement. Such Web clients will be able to transport and display Web pages composed of components that represent a myriad of different types of Web content and applications. Components consisting of Web content and applications will either be contained within a Web browser, a desktop or mobile containers that can be moved across the Web from one place to another. The implications with regard to the context of this chapter is that back-end applications such as database servers, GIS packages and DSS-related applications existing both locally and over the Web can exist as individual components for use by the client. These distributed objects existing within compound documents can be controlled[7], managed and integrated seamlessly on the desktop by the operating system or other applications designed to handle such documents.

In summary, this section covers a whole range of diverse yet related technologies and concepts that further advance the ongoing evolution of the Internet. How GIS users and developers use these emerging technologies will ultimately shape the future of the GIS industry.

6.4 Final remarks

This thesis explores some of the ways the Internet can be used as a more accessible means of obtaining geographic information. There seems to be a steady migration of GIS applications operating on a desktop environment to ones that are increasingly reliant upon the storage and processing obtained from distributed client/server networks (Berners-Lee et al., 1992b; Berners-Lee et al., 1994b). The advent of Web related technologies on existing client/server systems present new opportunities to transform existing GIS software into powerful tools for decision support, information delivery and information discovery. There are a number of very compelling advantages of using the Web to distribute geographic information. In addition the development of scalable database technology, improved network bandwidth/transmission rates, falling cost of storage devices and the powerful on-line query and analysis tool, the Web phenomenally reduces the project cost of creating and deploying very sophisticated software applications. These applications are more secure and cost-effective as well as easier to develop and maintain for a relatively large and distributed on-line community of users.

The software applications constructed and examined for this research portrays the Internet as the primary medium for the storage of geospatial data, analysis using GIS operations, and on-line display as digital maps. This can eventually include large federated systems whose constituent components are individually implemented on servers by different site developers located anywhere on the Web. The components can draw geospatial data (also other geographic information) from various sources, operate together by exchanging information, co-ordinate the processing of geospatial data in stages and interactively communicate or display the results to clients. The components of GIS software on this scale, separated over such large distances, require a great degree of co-ordination and standardisation. It is hoped that the technical and institutional issue examined in this thesis further clarifies the

[7] On the client side, the languages used to program these components (Java, JavaScript, Visual Basic, VB-Script etc) are well defined.

fundamental issues and trends that are responsible for the Internet becoming a new computing architecture for GIS.

References

Abbey, M. and Corey, M.J., 1995, *Oracle: A Beginner's Guide* (Berkeley: Osborne McGraw-Hill).

Abel, D.J., Kilby P.J. and Cameron M.A., 1994, A federated systems approach to design of spatial decision support systems. In *Proceedings of the 6th International Symposium on Spatial Data Handling,* (London: Taylor & Francis), pp. 46-59.

Anderson, J. T. and Stonebraker S., 1994, Metadata schema for satellite images. Research Report 94/58, SEQUOIA 2000 project, (Berkeley: University of California).

Anuff, E., 1996, *The Java Sourcebook* (New York: John Wiley & Sons).

Aronoff, S., 1989, *Geographic Information Systems: A Management Perspective* (Ottawa: WDL Publications).

Asproth, V., 1993, Trends in GIS research areas. In *Proceeding of the 4th European Conference and Exhibition on GIS,* (Genoa, Italy), pp. 17-24.

Bass, L. and Coutaz, J., 1991 *Developing Software for the User Interface* (Addison-Wesley).

Berners-Lee, T.J., 1992a, *Universal Resource Locators,* (Geneva: CERN).

Berners-Lee, T.J., Cailliau R., Groff J-F. and Pollermann, B., 1992b, World-Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy,* **2**(1), pp. 52-58.

Berners-Lee, T.J., Masinter, L. and McCahill, M., 1994a, Uniform Resource Locators (URL), [RFC 1738], CERN, Xerox Corporation, University of Minnesota.

Berners-Lee, T.J., Cailliau, R., Luotonen, A., Nielsen, H.F. and Secret, A., 1994b, *Communications of the ACM,* **37**(8), pp. 76-82.

Berson, A. and Smith, S.J., 1997, *Data Warehousing, Data Mining & OLAP* (New York: McGraw-Hill).

BIC Metadata Form, 1997, BIC Metadata Form: A Web form based means to submit metadata to a Clearinghouse, (Austein, Texas: Borderlands Information Center).

Brebner, G., 1997, *Computers in Communication* (London: The McGraw-Hill Companies).

Bresnahan, P.J., Cowen D.J, Ehler G.B., King E., Shirley W.L. and White T., 1994, Using geographical data browsers in a networked environment. In *Proceedings of the 6th International Symposium on Spatial Data Handling*, (London: Taylor & Francis), pp. 921-932.

Booch, G., 1994, *Object Oriented Analysis and Design with Applications* (Redwood City, California: Benjamin Cummings).

Burrough, P.A., 1986, *Principles of Geographical Information Systems for Land Resources Assessment* (Oxford, England: Clarendon Press)

Burrough, P.A., 1992 Development of intelligent geographical information systems. *International Journal of Geographical Information Systems,* **6**(1), pp. 1-11.

Calkins, H.W., 1992, Institutions sharing spatial information. In *Networking Spatial Information Systems,* edited by P.W. Newton, P.R. Zwart, M.E. Caveill (London and New York: Belhaven Press), pp. 283-293.

Camarata, Jr. S.J., 1992, The integration of new technologies and distributed architecture: A new revolution for GIS. In *Networking Spatial Information Systems,* edited by P.W. Newton, P.R. Zwart, M.E. Caveill (London and New York: Belhaven Press), pp. 49-62.

Cassettari, S., 1993, *Introduction to Integrated Geo-Information Management* (London: Chapman and Hall).

CEN, 1995, *Geographic Information - Data Description - Metadata (Draft)* (Brussels: Comité Européen de Normalisation).

Chen, J., Larson, R. and Stonebreaker, M., 1991, The Sequoia 2000 Object Browser. Research Report 91/4, Sequoia 2000 project, (Berkeley: University of California).

Chen, P.P., 1976, The Entity-Relationship Model – Toward a united view of data. *ACM Trans. Database Systems,* **1**(1), pp. 9-36.

Cheong F.C., 1996, *Internet Agents: Spiders, Wanderers, Brokers, and Bots* (Indianapolis: New Riders).

Circular A-16, 1994, Co-ordination of Surveying, Mapping, and Related Spatial Data Activities (revised). Executive Office of the President, (Washington D.C.: Government Printing Office).

Clinton, W., 1994, Co-ordinating Geographic Data Acquisition and Access: The National Spatial Data Infrastructure, Executive Order 12906, (Washington D.C.: The White House).

Coad, P. and Yourdon, E., 1991, *Object-Oriented Analysis* (Englewood Cliffs, US: Prentice Hall).

Coad, P. and Yourdon, E., 1992, *Object-Oriented Design* (Englewood Cliffs, US: Prentice Hall).

Corpsmet95, 1997, A metadata creation tool developed for the U.S. Army Corps of Engineers, (Middlefield CT: Coastal Oceanographics, Incorporated).

Cortese, A., 1995, The Software Revolution. *Business Week,* December, pp. 44-53.

Cuthbertson, M., Hamilton, M. and Monckton, C., 1994, Meta-Information, Internetworking and GIS, In *Proceeding of GISRUK'94,* (Leicester, UK: Leicester University), pp. 383-392.

Date, C.J., 1986 *An Introduction to Database Systems, Vol. I, 4th edition,* Addison-Wesley.

Daumiller, G., 1996 Data Dictionary: Intelligent CSDGM metadata creation tool for Arc/Info, (Montana, US: Montana State Library).

DeBra, P. and Post, B., 1994, Information retrieval in the World Wide Web: Making client-based search feasible. In *Proceeding of the First International World Wide Web Conference,* edited by R. Cailliau, O. Nierstrasz, M. Ruggier (Geneva).

DEC C, 1994, *Run-Time Library Reference Manual for OpenVMS Systems,* (Maynard, Massachusets: Digital Equipment Corporation).

DEC C, 1996, *User's Guide for OpenVMS Systems* (Maynard, Massachusets: Digital Equipment Corporation).

Deep, J. and Holfelder, P., 1996, *Developing CGI Applications with Perl* (New York: John Wiley & Sons)

Deitel, H.M. and Deitel, P.J., 1994, *C: How to Program 2nd edition* (London: Prentice-Hall International)

Deitel, H.M. and Deitel, P.J., 1998, *Java: How to Program 2nd edition* (London: Prentice-Hall International)

Denenberg, R., Kunze, J. and Lynch, D., 1996, Uniform Resource Locators for Z39.50, November [RFC 2056], Library of Congress, University of California at San Francisco, Silver Platter Information Ltd.

Dickinson, H.J. and Calkins, H.W., 1988, The ecnonomic evaluation of implementing a GIS. *International Journal of Geographical Information Systems,* **2**(4), pp. 307-327.

Dix, A., Finlay, G. and Beale, R., 1993, *Human-Computer Interaction* (Prentice-Hall)

Dodd, S., 1982, Cataloging machine-readable data files, (Chicago: American Library Association).

Dowers, S., 1995, CONVGIF: Converts files to VMS compatible format when transferred from DOS/Windows systems, (Edinburgh: Department of Geography, University of Edinburgh).

Dozier, J., 1991, Early EOSDIS: A data and information system for the study of global change. Research Report 91/8, Sequoia 2000 project, (Berkeley: University of California).

Dozier, J., 1992, How Sequoia 2000 addresses issues in data and information systems for global change. Research Report 92/14, Sequoia 2000 project, (Berkeley: University of California).

Douglas, S.A. and Tomilinson, R.F., 1992, Assessing costs and benefits of geographical information systems: methodological and implementation issues. *International Journal of Geographical Information Systems,* **6**(3), pp. 247-256.

Eichmann, D., 1994, Ethical Web Agents. In *Proceedings of the Second International World-Wide Web Conference,* October, Chicago, US.

ESRI, 1991, *ARC/INFO® Users's Guide: Data Conversion* (Redlands, California: Environmental Systems Research Institute).

ESRI, 1994a, *ARC Macro Language: Developing ARC/INFO® menus and macros with AML ™* (Redlands, California: Environmental Systems Research Institute).

ESRI, 1994b, *ARC/INFO® User's Guide: ARC/INFO® Data Model, Concepts, & Key Terms* (Redlands, California: Environmental Systems Research Institute).

ESRI, 1995, ESRI's approach to interoperability, A white paper, (Redlands, California: Environmental Systems Research Institute).

ESRI, 1997, MapObjects™: Embeddable GIS and mapping components for application developers, An introductory pamphlet, (Redlands, California: Environmental Systems Research Institute).

ESRI, 1998a, ArcExplorer™: GIS Data Viewer, (Redlands, California: Environmental Systems Research Institute).

ESRI, 1998b, ArcView Internet Map Server™: Publish Your ArcView GIS Maps on the World Wide Web!, (Redlands, California: Environmental Systems Research Institute).

Etzioni, O. and Weld, D., 1994, A softbot-based interface to the Internet. *Communications of the ACM,* **37**(7), pp. 72-76.

Federal Geographic Data Committee, 1994, *Content standards for digital spatial metadata*, (Washington D.C.: Federal Geographic Data Committee).

Fegeas, R.G., Cascio J.L. and Lazar R.A., 1992, An overview of FIPS 173: The Spatial Data Transfer Standard. *Cartography and Geographic Information Systems,* **19**(5), pp. 278-293.

Ferrari, D., Pasquale, J. and Polyzos, G.C., 1991, Network Issues for Sequoia 2000. Research Report 91/6, (Berkeley: University of California).

FGDC, 1992, *Information Exchange Forum,* (Washington D.C.: Federal Geographic Data Committee).

FGDC, 1994, *The 1994 Plan for the National Spatial Data Infrastructure: Building the foundation of an information based society,* (Washington D.C.: Federal Geographic Data Committee).

FGDC, 1995, *Overview-Content Standards for Digital Geospatial Metadata,* (Washington D.C.: Federal Geographic Data Committee).

Fisher, P.F., 1991, Spatial data sources and data problems. In *Geographical Information Systems Vol 1: Priciples,* edited by D.J. Maquire, M.F. Goodchild, D.W. Rhind (London: Longman), pp. 175-189.

Flanagan, D., 1997, *JavaScript: The Definitive Guide 2nd edition* (Cambridge, UK: O'Reilly & Associates).

Flanagan, D., 1997, *Java in a NutShell: A desktop quick reference* (Cambridge, UK: O'Reilly & Associates).

Flowerdew, R., 1991, Spatial data integration. In *Geographical Information Systems Vol 1: Priciples,* edited by D.J. Maquire, M.F. Goodchild, D.W. Rhind (London: Longman), pp. 375-387.

Frank, S.M., 1994, *Cataloging Paradigms for Spatial Metadata,* Unpublished Ph.D. thesis, (Maine: Department of Surveying Engineering, University of Maine).

Genesereth, M.R. and Ketchpel, S.P., 1994, Software Agents, *Communications of the ACM,* **37**(7), pp. 48-53,147.

GENIE project, 1995, *General Information*, (England UK: Midlands Regional Laboratory and Loughborough University of Technology).

GeoMedia™, 1997,: Communicating Geographically on the Web, An introductory pamphlet, (Huntsville, Alabama: Intergraph Corporation).

Gittings, B.M. and Healey, R.G., 1989, *An Introduction to the ORACLE Relational Database Management System,* (Edinburgh: Department of Geography, University of Edinburgh).

Gittings, B.M. and Towers, A.L., 1995, Earthquake analysis through global information networks. *GIS ASIA/PACIFIC,* **1**(2), pp. 28-31.

Goodchild, M., 1995, Information highways. In *The AGI source book*, (London: The Association for Geographic Information), pp. 45-55.

Goucher, G. and Mathews J., 1994, *NASA Tech Brief: Common Data Format (CDF)*, (Greenbelt, MD: National Space Science Data Center).

Graham, I.M., 1995, *Object Oriented Methods* (Wokingham, England: Addison-Wesley).

Graham, I.M., 1996, *HTML Source Book, 2nd edition* (New York: John Wiley & Sons).

Green, D.R. and Lindsey J.M., 1992, Turning data into information: Assessing GIS user interfaces. In *Proceedings of the Annual Conference of the Association of Geographic Information,* (Birmingham, UK: AGI).

Greenberg, G.M., 1992 *AUTODXF.AML program,* (Florida, US: St. Johns River Water Management District).

Gundavaram, S., 1996, *CGI Programming on the World Wide Web* (Cambridge, UK: O'Reilly & Associates)

Guptill, S.C., 1992, The SEQUOIA 2000 approach to managing large spatial object databases. In *Proceedings of the 5th International Symposium on Spatial Data Handling,* Charleston, South Carolina, US, pp. 642-651.

Guptill, S.C., 1994, Synchronization of discreet geospatial data bases. In *Proceedings of the 6th International Symposium on Spatial Data Handling,* (London: Taylor & Francis), pp. 945-956.

Hakala, J., 1996, *Z39.50-1995: Information retrieval protocol,* (Helsinki: Automation Unit of Finnish Research Libraries, Helsinki University Library).

Hansen, D., 1997, MetaMaker 2.10 or 2.20: A CSDGM metadata creation tool written in Microsoft Access, (Environmental Management Technical Center, USGS Biological Resources Division).

Harvey, R.A., 1996, *Development of a World Wide Web Interface for the Dissemination of ScotLIS Land and Property Information,* Unpublished masters thesis, (Edinburgh: Department. of Geography, University of Edinburgh).

Healey, R., Dowers, S., Gittings, B. and Mineter, M., 1998, *Parallel Processing Algorithms for GIS* (London: Taylor & Francis).

Hickman, C. and Altheide P., 1996, *Overview of SDTS profiles,* (Rolla, Missouri: United States Geological Survey).

Howe, D. R., 1989, *Data Analysis for Data Base Design* (London: Edward Arnold).

Jepson, B., 1997, *JAVA Database Programming* (New York: John Wiley & Sons).

Jones, D., 1995, Fetch_HTTP: A Program that connects to an HTTP server and perform a fetch of the URL, (Columbus, Ohio: Ohio State University).

Katz, R.H., 1991, SEQUOIA 2000: High Performance Network and Channel-Based Storage. Research Report 91/2, SEQUOIA 2000 project, (Berkeley: University of California).

Katz, S., 1995, SDTS geospatial data conversion utilities, (Washington D.C.: US Bureau of Land Management).

Katz, S., 1996, BLMDOC7.AML: A metadata extraction utility for UNIX ARC/INFO®, (Washington D.C.: US Bureau of Land Management).

Kautz, H.A., Selman, B. and Coen, M., 1994, Bottom-Up Design of Software Agents. *Communications of the ACM*, **37**(4), pp. 143-147.

Kuhn, W., 1994, Defining semantics for spatial data transfers. In *Proceedings of the 6th International Symposium on Spatial Data Handling,* (London: Taylor & Francis), pp. 973-987.

Kunze, J.A., 1995, *Basic Z39.50 server concept and creation,* (Berkeley: University of California and U.S. National Library of Medicine).

Lanter, D.P., 1991, Design of a Lineage-Based Meta-Data Base for GIS. *Cartography and Geographic Information Systems,* **18**(4), 255-261.

Lanter, D.P. and Surbey, C., 1994, Metadata analysis of GIS data processing: A case study. In *Proceedings of the 6th International Symposium on Spatial Data Handling,* (London: Taylor & Francis), pp. 314-324.

Larson, R. R., 1994, SEQUOIA 2000: Design and Development of a Network-Based Electronic Library. Research Report 94/54, SEQUOIA 2000 project, Berkeley, University of California.

Laurini, R., 1995, Distributed geographic databases: an overview. In *The AGI source book*, (London: The Association for Geographic Information), pp. 45-55.

Laurini, R. and Thompson, D., 1992, *Fundamentals of Spatial Information Systems* (London: Academic Press).

Levinson, E., 1997, Content-ID and Message-ID Uniform Resource Locators, [RFC 2111], XIson Inc.

Linsey, T.K. and Raper, J.F., 1993 HyperArc: a task-oriented hypertext GIS interface. *International Journal of Geographical Information Systems,* **7**(5), pp. 435-452.

Mace, S., Flohr, U., Dobson, R. and Graham, T., 1998, Weaving a better Web, *Byte Magazine,* **23**(3), pp. 58-68.

Machine-Readable Bibliographic Information Committee, 1996, *The USMARC Formats: Background and Principles*, (Washington D.C.: Library of Congress).

MapInfo ProServer, 1998, An inside look at MAPINFO ProServer, (Troy, NY: MapInfo Corporation).

Mapping Science Committee, 1993, *Towards a Co-ordinated Spatial Data Infrastructure for the Nation* (Washington D.C.: National Academy Press).

Massam, P.S., 1994, *Achieving Global Access to Geographic Information using the World-Wide Web,* Unpublished masters thesis, (Edinburgh: Department. of Geography, University of Edinburgh).

Masser, I. and Campbell, H., 1993, Information Sharing and the implementation of GIS: Some key issues. In *Proceedings of GIS Research UK,* (Keele UK: Keele University), pp. 112-121.

McClanahan, D., 1996, *Oracle Developer's Guide* (Berkeley: Osborne McGraw-Hill).

McClure, C.R. and Moen, W.E., 1994, *Using Z39.50 in an application for the Government Information Locator Service (GILS): A Background Paper,* (Syracuse, NY: Center for Science and Technology, Syracuse University).

Medyckyj-Scott, D., Cuthbertson, M. and Newman, I., 1996 Discovering environmental data: metadatabases, network information resources tools and the GENIE system. *International Journal of Geographical Information Systems,* **10**(1), pp. 65-84.

Medyckyj-Scott D. and Morris B., 1998, The virtual map library: Providing access to OS digital map data via the WWW for the Higher Education Community, (Edinburgh: Data Library, University of Edinburgh).

Mockapetris, P., 1987a, Domain Names—Concepts and Facilities, November [RFC 1034].

Mockapetris, P., 1987b, Domain Names—Implementation and Specification, November [RFC 1035].

Monckton, C., 1993, Meta-information and future GIS. In *Proceedings of the Annual Conference of the Association of Geographic Information,* (Birmingham, UK: AGI).

Moore, A., Parr, G., McCormick, D. and McAlister, S., 1999, Integrated Regional Development Support System (IRDSS). In *Innovations in GIS 6: A Data and Technology Infrastructure for GIS,* edited by B.M. Gittings (London: Taylor & Francis).

Murnion, S. and Munroe, G., 1997, Surfing on the 'Net' Information Resources for GIS Users, (Belfast, Northern Ireland: Queen's University).

Naik, D.C., 1998, *Internet standard and protocols* (Redmond, Washington: Microsoft Press).

NCGIA, 1995, NCGIA Research Initiatives, (Santa Barbara: National Center for Geographic Information and Analysis, University of California).

Nebert, D., 1994, Metamorph 1.0: On-line CSDGM metadata entry tool, (Reston, Virginia: United States Geological Survey).

Nebert, D., 1995a, *Serving Digital Map Information through the World-Wide Web and Wide-Area Information Server Technology,* , (Reston, Virginia: United States Geological Survey).

Nebert, D., 1995b, *Status of the National Geospatial Data Clearinghouse on the Internet,* (Reston, Virginia: United States Geological Survey).

Nebert, D. and Fullton, F., 1994, *Use of Z39.50 to search and retrieve geospatial data,* (Project Alexandria, University of California at Santa Barbara and Clearinghouse for Networked Information Discovery and Retrieval).

Negri, N., 1996, UNIX DOCUMENT.AML, Version 7.03 and 7.04: Utility to extract metadata from an ARC/INFO® coverage, (Reston, Virginia: United States Geological Survey).

Nelson, D., 1997, fgdcmeta.aml, Version 1.1: Generates metadata files in ASCII text from ARC/INFO® geospatial data sets in a UNIX or NT environment, (Illinois State Geological Survey).

Newman, C. and Myers, J., 1997, ACAP -- Application Configuration Access Protocol, November [RFC 2244], Innosoft, Netscape.

Newman, C., 1997, IMAP URL Scheme, [RFC 2192], Innosoft.

Newton, P.W, Zwart, P.R. and Cavil, M.E., 1992a, Introducing the next wave: Networking spatial information systems. In *Networking Spatial Information Systems,* edited P.W. Newton, P.R. Zwart, M.E. Caveill (London and New York: Belhaven Press), pp. 1-11.

Newton, P.W, Zwart, P.R. and Cavil, M.E., 1992b, Inhibitors and facilitators in high speed networking or spatial information systems. In *Networking Spatial Information Systems,* edited by P.W. Newton, P.R. Zwart, M.E. Caveill (London and New York: Belhaven Press), pp. 251-263.

NGDF, 1999, *Discovery Metadata Guidelines Version 1.1,* (Southhampton, UK: National Geographic Data Framework Management Board, Ordnance Survey).

Niederst, J. 1999, *Web Design in a Nutshell* (Cambridge: O'Reilly).

Northcutt R.T. and Shein K.A., 1998, A Java Based Query Preview Prototype for Directory Search and Refinement, (Greenbelt, MD: Global Change Master Directory).

Ogilvie, J., 1995, Pythia technical information, Appian Interactive Corporation.

Olsen, L., 1996, *GCMD System Documentation (Software Version 5.0),* (Global Change Data Center, NASA/Goddard Space Flight Centre).

OpenGIS®, 1996, *The OpenGIS Abstract Specification: An Object Model for Interoperable Geoprocessing*, Revision 1, Project Document Number 96-015R1, (Wayland, MA: Open GIS Consortium Incorporated).

Oracle®, 1989, *Pro*C User's Guide Version 1.1* (Redwood City, CA: Oracle Corporation).

Oracle®, 1992, *Programmer's Guide to the Oracle® Precompilers Version 1.5* (Redwood City, CA: Oracle Corporation).

Oracle, 1999a, Oracle8i™: Release 2 news features summary, (Redwood City, CA: Oracle Corporation).

Oracle, 1999b, Oracle8i™ Spatial: Features overview, (Redwood City, CA: Oracle Corporation).

Oracle, 1999c, Oracle8i™ Spatial: Data sheet, (Redwood City, CA: Oracle Corporation).

Oracle, 1999d, Using Oracle8i™ interMedia for location queries and data warehousing, (Redwood City, CA: Oracle Corporation).

Oracle, 1999e, Oracle8i™: The database for Internet computing, (Redwood City, CA: Oracle Corporation).

Oracle, 1999f, Spatially enabling the enterprise: An Oracle business white paper , (Redwood City, CA: Oracle Corporation).

ORACLE7™ Server, 1992a, *SQL Language Reference Manual* (Redwood City, CA: Oracle Corporation).

ORACLE7™ Server, 1992b, *Concept Manual* (Redwood City, CA: Oracle Corporation).

Orfali, R., Harkey, D. and Edwards, J., 1996a, *The Essential Client/Server Survival Guide* (New York: John Wiley & Sons).

Orfali, R., Harkey, D. and Edwards, J., 1996b, *The Essential Distributed Object Survival Guide* (New York: John Wiley & Sons).

Orlowska, M.E., Lister A.M. and Frogg I., 1992, Decentralising spatial databases. In *Networking Spatial Information Systems,* edited P.W. Newton, P.R. Zwart, M.E. Caveill (London and New York: Belhaven Press), pp. 63-76.

Owen, J., 1993, Data interchange? A personal view of today's technology. In *Proceedings of the 4th European Conference and Exhibition on GIS,* Genoa, Italy, pp. 25-29.

Oxborrow, E., 1989, *Databases and Databases Systems: Concepts and Issues 2nd edition* (Canterbury, UK: Chartwell-Bratt).

Pasquale, J., Fall, K. and Forrest, J., 1992, SEQUOIA 2000: Network (S2Knet) Handbook. Research Report 92/19, SEQUOIA 2000 project, San Diego, University of California.

Plewe, B., 1994, The GeoWeb Project: Using WAIS and the World Wide Web to aid location of Distributed Data Sets, (Buffalo: State University of New York).

Plewe, B., 1997, So you want to build an online GIS, *GIS World*, **10**(11), pp. 58-60.

Pradhan, A. and Gittings, B.M., 1996, Accessing spatial data and metadata over the Internet. *GIS Research UK (GISRUK) Conference* (Canterbury: University of Canterbury).

Pradhan, A. and Gittings, B.M., 1999, Web based geospatial data conversion tools. In *Innovations in GIS 6: A Data and Technology Infrastructure for GIS,* edited by B.M. Gittings (London: Taylor & Francis).

Pollock, R. J. and McLaughlin, J. D., 1990, Data-base management system technology and Geographic Information Systems. *Journal of Surveying Engineering,* **117**(1), pp. 9-26.

Pretty, R.W., 1992, LANS, MANS and WANS: Introductory tutorial on computer networking. In *Networking Spatial Information Systems* edited P.W. Newton, P.R. Zwart, M.E. Caveill (London and New York: Belhaven Press), pp. 15-27.

Putz, S., 1994, Interactive information services using World Wide Web hypertext. *Computer Networks and ISDN Systems,* **27**(2), pp. 273-280.

Rainio, A., 1995, *EDI-Based Information Service.* (Helsinki, Finland: Finnish Public Administration Recommendations).

Rainio, A. and Ahonen, P., 1994, EDI and joint use of Geographic Information. In *Workshop 3 -17th Urban Data Management Symposium,* Helsinki-Espoo, Finland.

Randall, J. and Randall, N., 1995, *The World Wide Web Unleashed* (Indianapolis, Indiana: Sams.net Publishing).

Raper, J.F. and Bundock, M.S., 1990, GIS user interfaces: A window on the future. In *Proceedings of the Annual Conference of the Association of Geographic Information,* (Birmingham, UK: AGI).

Rhind, D., 1992, Data access, charging and copyright and their implications for geographic information system. *International Journal of Geographical Information Systems,* **6**(1), pp. 13-30.

Riechen, D., 1994, Intelligent agents: Introduction to Special Issue. *Communications of the ACM,* **37**(7), pp. 18-21.

Robinson, A. H., Randall S. D., Morrison J. L. and Muehrcke P. C., 1984, *Elements of Cartography Fifth Edition* (New York: John Wiley & Sons).

Safe Software Inc., 1999, *FME: Reference Manual Version 2.3* (Surrey, BC: Saif Software Inc.)

SAIF, 1994, *SAIF: Formal Definition Release 3.1.,* (Victoria, BC: MAPS BC Surveys and Resource Mapping Branch).

Schweitzer, P., 1996a, *FGDC Metadata FAQ*, (Reston, Virginia: United States Geological Survey).

Schweitzer, P.N., 1996b, mp (Metadata Parser) Version 2.0: A utility to validate metadata against the CSDGM (1998) and convert it to standardised text, HTML, SGML, or DIF form, (Reston, Virginia: United States Geological Survey).

Schweitzer, P.N., 1996c, cns (Chew and Spit) Version 2.0: metadata format pre-processor for mp (CSDGM 1998 compliant), (Reston, Virginia: United States Geological Survey).

Schweitzer, P.N., 1998, xtme Version 2.0: X toolkit metadata editor intended to simplify the process of creating standardised metadata, (Reston, Virginia: United States Geological Survey).

SDTS, 1994, *Spatial Data Transfer Standard. US Federal Information Processing Standard (FIPS) 173,* (Reston, Virginia: SDTS Maintenance Authority, United States Geological Survey).

SDTS Task Force, 1996, *The Spatial Data Transfer Standard: Handbook for Technical Staff,* Federal Systems Integration and Management Center (FEDSIM), (Reston, Virginia: United States Geological Survey).

Shneiderman, B., 1992, *Designing the User Interface* (Addison-Wesley).

Sondheim, M., Gardels, K. and Buehler, K., 1998, GIS Interoperability, (Berkeley: Center for Environmental Design Research, University of California).

Sommerville, I., 1996, *Software Engineering Fifth Edition* (Harlow, England: Addison-Wesley Publishing Company).

Stonebraker, M., 1991, An overview of the SEQUOIA 2000 project. Research Report 91/5, SEQUOIA 2000 project, (Berkeley: University of California).

Stonebraker, M., 1993, Large object support in POSTGRES. Research Report 93/30, SEQUOIA 2000 project, (Berkeley: University of California).

Stonebraker, M., 1994, SEQUOIA 2000 - A reflection on the first three years. Research Report 94/58, SEQUOIA 2000 project, (Berkeley: University of California).

Stonebraker, M. and Dozier, J., 1991, Large capacity object servers to support global change research. Research Report 91/1, SEQUOIA 2000 project, (Berkeley: University of California).

Stonebraker, M., Frew, J. and Dozier, J., 1993, The Sequoia 2000 architecture and implementation strategy. Research Report 93/23, SEQUOIA 2000 project, (Berkeley: University of California).

Strand, E.J., 1998, GIS users watch, wander and wait for the World Wide Web's Evolution. *GIS World*, March **11**(3), pp. 32-34.

Sun Microsystems, 1995, *The Java language: A white paper,* (Mountainview, CA: Sun Microsystems).

Thomas, S. A., 1996, *IPng and TCP/IP Protocols: Implementing the next generation Internet,* (New York: John Wiley & Sons).

Towers, A.L., 1993, *Earthquake Monitoring and Prediction: A case study of GIS data integration using the internet,* Unpublished masters thesis, (Edinburgh: Department of Geography, University of Edinburgh).

Towers, A.L. and B.M. Gittings, 1995, Earthquake monitoring and prediction. In *innovations in GIS 2,* edited by P.F. Fisher (London: Taylor and Francis).

Veizades, J., Guttman, E., Perkins, C. and Kaplan, A., 1997, Service Location Protocol, [RFC 2165], @Home Network, Sun Microsystems.

Vogal, R., 1996, *Directory Interchange Format (DIF) Writer's Guide, Version 5.0a,* Global Change Master Directory (NASA), http://gcmd.gsfc.nasa.gov

Waker, D.R.F., Newman, I.A., Medyckyj-Scott, D.J. and Ruggles, C.L.N., 1992 A system for identifying datasets for GIS users. *International Journal of Geographical Information Systems,* **6**(6), pp. 511-527.

Weider, C., Reynolds J. and Heker S., 1992a, Executive Introduction to Directory Services Using the X.500 Protocol, [RFC 1308].

Weider, C., Reynolds J. and Heker S., 1992b, Technical Overview of Directory Services Using the X.500 Protocol, [RFC 1309].

Yeager, N.J. and McGrath R.E., 1996. *Web Server Technology: The advanced guide for World Wide Web information providers* (San Francisco, California: Morgan Kaufmann Publishers).

Z39-50, 1992, *Information Retrieval Services and Protocol: American National Standard, Information Retrieval Application Service Definition and Protocol Specification for Open Systems Interconnections,* ANSI/NISO.

Z39-50, 1995, *Information Retrieval Services and Protocol,* ANSI/NISO.

APPENDIX A

The GMS compact disk

The amount of code that makes up the Geospatial Metadata Server is far too excessive to present in typed text. The entire system consists of 275 files including files containing program code (ANSI C, Pro*C, HTML, DCL, AML and SQL), text files and images contained within 10 folders or directories. Therefore, a CD-ROM is provided with this dissertation to allow readers to more easily manoeuvre through the file directories and view the program code for each of the GMS utilities. Only the program code for the entire GMS package is contained on the CD-ROM within the directory labelled *Code for GMS* and the code for each component of the system is in its own directory. The HTML pages within these directories are all relatively linked. One can begin by viewing the file *GMS home page.html* and browse through all the HTML documents of each GMS component using a Web browser. The other top-level directories on the CD-ROM contain the Web pages, program code and executable files as they are implemented on the Department of Geography Web server. The WWW directory contains mostly Web pages, text-files and images. The SCRIPTS directory contains most of the program code related with CGI scripts. The ORACLE directory contains the program code used to implement the GMS metadatabase.



APPENDIX B

Interchange standards

Spatial Data Transfer Standard (SDTS)

The purpose of the Spatial Data Transfer Standard (SDTS) is to provide a practical and comprehensive means to transfer geospatial data between different spatial analytical software packages so that such systems can use spatial data in their native formats. SDTS is a Federal Information Processing Standard (FIPS 173) for the non-proprietary, vendor neutral distribution of geospatial data and for the exchange of geospatial data between dissimilar digital geographic information systems SDTS, 1994). It is a self-contained standard that includes spatial data, attributes, geo-referencing, data quality report, data dictionary, and other supporting transfer related metadata. Producing geospatial data with SDTS is more a process by which successive "levels" are defined by initially specifying real world entities and then moving down to the detailed aspect of geocoding. There are three levels:

Conceptual Level - the geometric and topological characteristics and relationships of real world entities.
Logical Level - construction of a data model for geocoding purposes.
Physical Level - rules and specifications for geocoding data on a particular medium.

Part 1 or logical specification deals with the first two levels which include the conceptual model specifying the spatial object types, the data quality report and the SDTS module layout. Spatial features in part 2 of the standard provides greater compatibility during data transfers by defining common spatial feature terms that are catalogued along with associated attributes. Spatial features are concerned with the logical level of the specification where there are many possibilities for the

structuring, naming, and placement of information into modules and fields (SDTS Task Force, 1996).

The ISO 8211 is and internationally accepted standard that specifies the physical transfer of data by encoding digital information on storage media or for electronic transmission. The information is arranged during transfers in a standard fashion so that it can be encoded and decoded easily. The physical level of the SDTS specification (part 3 of the standard) encodes SDTS transfers on hardware storage devices (i.e. disk drives) and communication lines.

Data-conversion to and from SDTS and other data formats is a daunting task, keeping in mind the shear number of spatial objects that exist even among the vendor spatial geospatial data formats. To make the task of writing translation software more manageable, SDTS is sub-divided into "profiles" that concentrate on the characteristics and properties of a specific data model. Profiles categorise and group certain types of SDTS spatial objects and related SDTS modules and indicate if they are required, allowed or prohibited objects with respect to a particular implementation (Hickman and Altheide, 1996). Two profiles that are part of the SDTS specification include the Topological Vector Profile (TVP) or FIPS 173 and the Raster Profile. TVP is used to transfer vector data such as USGS DLG or US Census Bureau TIGER files that exhibit geometry and topology whose basic elements consist of point, line, area and attribute information. The Raster Profile specifies data transfers whose features are represented as byte maps. An example is the USGS's DEM data sets.

Common Data Format (CDF)

The Common Data Format (CDF) is a conceptual data abstraction that facilitates the storing and manipulation of generic data ensembles from a wide variety of scientific disciplines as multi-dimensional array data sets. The programming interface provides an abstract view of the CDF data model which is much simpler than dealing with the actual physical file format and is independent of any particular device. The primary advantage of the high level of abstraction in the CDF data model

is that the file need not be physically formatted in order to extract the data because such low-level operations are performed by the interface.

CDF is independent through an internal format containing two types of data that includes a data dictionary (metadata) provided by the user and actual data values. It is the self-describing property of the format that allows CDF to be used in a multitude of disciplines. The interface is capable of accessing both types of data through a series of C language libraries and toolkits as well as some FORTRAN source files. The interface has been ported to a number of platforms allowing the data independent properties of CDF to be used by a wide variety of software applications. Data independence allows these applications to perform different operations on different data sets because the interface frees the application from having to understand the underlying physical file format. Because the format already contains its own metadata, there can be no ambiguity between data-providers and users when describing any particular data file.

The multi-dimensional arrays or regular grids that comprises the CDF data model store data from observable parameters by assigning each to a particular variable. Such a model allows related software applications to slice data across multi-dimensional subspaces, access entire structures of data, perform sub-sampling of data or access a single element independent of its relationship to other elements (Goucher and Mathews, 1994). The interface can access multiple variables thereby allowing applications to work with multiple data sets regardless of their structures or sizes. Since the interface is developed using C libraries and FORTRAN source code it can easily be incorporated into various analysis utilities.

Spatial Archive and Interchange Format (SAIF)

SAIF is a geospatial data standard developed by the Survey and Resource Mapping Branch of British Columbia Environment, Lands and Parks as a means of sharing spatial and spatial-temporal data. The standard is an ideal non-proprietary format for the distribution of geographic data because it is designed to adhere to two existing international standards, the ISO SQL Multimedia spatial standard and Open Geodata

Interoperability Standard (OGIS). The ISO SQL/MM standard specifies how spatial and temporal information are addressed by databases and is currently implemented in SQL3, the latest specification of the Sequential Query Language. OGIS will eventually enable different GIS applications to communicate through its own Virtual Geodata Model and Application Programming Model (OpenGIS®, 1996).

SAIF follows a multiple inheritance, object oriented paradigm. Three levels of abstraction are used. The most abstract level consists of mathematical constructs such as tuples, lists, sets, enumerations and primitives. From these fundamental constructs a number of spatial and temporal constructs can be specified which constitutes the second level of abstraction. Finally, the last level is the specification of geographic objects from second level spatial and temporal constructs. These geographic objects specify real world phenomenon and are named as such. There are over 300 spatial and temporal constructs that constitute the second layer of abstraction within the SAIF standard schema and the bulk of the standard is devoted to describing these generic constructs. So just about any type of third level of abstraction or class (which can be 'instantiated' into a geographic object upon implementation) can be specified using these generic constructs. If additional spatial and temporal constructs are need then one must define them according to the first level of abstraction using mathematical constructs (SAIF, 1994).

Unlike other geospatial data transfer standards, SAIF not only supports the interchange of data between systems but also provides facilities for data storage. SAIF uses a Class Syntax Notation (CSN) as a data definition language for a spatial storage system (Healey et al., 1998). CSN was developed in order to define mathematical, spatial and temporal constructs using a text-based syntax. Because CSN is written in ASCII it can be parsed and read into a computer program. SQL or ASN.1 in their current version could not be used because it does not support abstract data types. Object oriented languages tend to be too complex and lack facilities for data management. CSN supports abstract data types that allow the encoding of instances of classes or geographic objects with their associated data. The encoding can be in ASCII or binary format for either a data stream or file. Developers can read the ASCII form since it follows the Object Syntax Notation (OSN) which is

compatible with CSN and together they form a data definition language known as SAIFtalk.

To transfer SAIF compliant data from one GIS or DBMS system to another is performed through a file format call SAIF/ZIP. This exchange file format typically contains the data model or format definition of the transferred data, associated metadata and geometric/textual feature descriptions. Like the GMS data-conversion utilities described in chapter 4, geospatial data from one or more vendor specific formats can be converted to one another using SAIF as a common data format. However it should be said the same is true for any of the geospatial data transfer standards examined in this section as well as any of the proprietary data format (Arc/Info's E00 interchange file format was used for the GMS conversion utilities). How well the data is translated or converted from one spatial data format to another depends on the translator software used. Either the FMEBC or FME software carries out Geospatial data translation to and from the SAIF format. FMEBC is freeware and is owned by the government of British Columbia; FME is a commercial version of the software marketed by the Vancouver based company Safe Software.

Neutral Transfer Format (NTF)

NTF was developed by the British Standards Institute (BSI) in May of 1992 and is referred to as British Standard 7567 for the encoding and transfer of digital data over communication networks. There are three parts to the standard:

- the data models and data structures used in the standard;
- ASCII form implementation of NTF for data transfer;
- the implementation of NTF using the international standard ISO/IEC 8211:1994[1] on information interchange.

The NTF standard has five levels. The fourth level describes topology such as how features are mapped to real world phenomenon, the relationships between features

[1] The full title of ISO 8211 is "Specification for a Data Descriptive File for Information Interchange". ISO 8211 is a concrete syntax and encoding rule standard developed to enable the fully automated interchange and archiving of file-oriented information. It is used as a transport mechanism by various geographic information transfer standards.

and how they are linked to the underlying geometry such as spatial location and extent. The fifth level provides specification to define existing formats and new record types. Data quality information can be specified for each of these levels by describing attributes, classifying features and listing coded records.

APPENDIX C

Attribute assignment for the E-R model

Identification Information
ii_id (Primary Key)
Abstract
Purpose
Currentness Reference
Progress
Maintenance and Update Frequency
Access Constraints
Use Constraints
Data Set Credit
Native Data Set Environment
Top Left Bounding Co-ordinate
Top Right Bounding Co-ordinate
Bottom Left Bounding Co-ordinate
Bottom Right Bounding Co-ordinate
Logical Consistency Report
Completeness Report
Percent Cloud Cover
access#
full name
password
ii_ci_id (foreign key)
ii_mri_id (foreign key)
ii_di_id (foreign key)
ii_sdi_id (foreign key)

Citation Information
ci_id (Primary Key)
Originator
Publication Date
Publication Time
Title
Edition
Geospatial Data Representation
Series Name
Issue Identification

Publisher
Publication Place
Online Linkage
Other Citation Details
access#

Source Information
si_id (Primary Key)
Source Scale Denominator
Type of Source Media
Source Currentness Reference
Source Citation Abbreviation
Source Contribution
access#
si_ci_id (foreign key)
si_ii_id (foreign key)

Spatial Data Information
sdi_id (Primary Key)
Spatial Reference Method
Other Spatial Reference Method
Latitude Resolution
Longitude Resolution
Geographic Co-ordinate Units
Map Projection Name
Other Projection's Definition
Grid Co-ordinate System Name
UTM Zone Number
UPS Zone Identifier
SPCS Zone Identifier
ARC System Zone Identifier
Other Grid System's Definition
Local Planar Description
Local Planar Georeference Information
Local Description
Local Georeference Information
Altitude Datum Name
Altitude Resolution
Altitude Distance Units
Altitude Encoding Method
Depth Datum Name
Depth Resolution
Depth Distance Units
Depth Encoding Method
Entity and Attribute Overview

access#

Distribution Information
di_id (Primary Key)
Resource Description
Distribution Liability
Custom Order Process
Technical Prerequisites
access#
di_cti_id (foreign key)

Digital Transfer Information
dti_id (Primary Key)
Format Name
Format Version Number
Format Version Date
Format Specification
Format Information Content
File Decompression Technique
Transfer Size
Network Resource Name
Network Address
Access Instructions
Host Operating System
Fees
Ordering Instructions
Turnaround Time
access#
dti_di_id (foreign key)

Metadata Information
mri_id (Primary Key)
Metadata Title
Metadata Date
Metadata Review Date
Metadata Future Review Date
Metadata Standard Name
Metadata Standard Version
Metadata Time Convention
Metadata Access Constraints
Metadata Use Constraints
access#
mri_cti_id (foreign key)

Contact Information
cti_id (Primary Key)
Contact Person
Contact Organisation
Contact Position
Address Type
Address
City
State or Province
Postal Code
Country
Contact Voice Telephone
Contact Facsimile Telephone
Contact Electronic Mail Address
Hours of Service
Contact Instructions
access#

Browse Graphic
bg_id (Primary Key)
BG File Name
BG File Description
BG File Type
BG File Online Linkage
access#
bg_ii_id (foreign key)

Date Time Period
dtp_id (Primary Key)
Calendar Date
Time of Day
Date Begin
Date End
Time Begin
Time End
access#

SI-DTP Map
dtp_id
si_id

II-DTP Map
dtp_id
ii_id

DI-DTP Map
dtp_id
di_id

Keyword Temporal
key_temporal_id (Primary Key)
Temporal Keyword
access#

Keyword Temporal Map
ii_id
key_temporal_id

Keyword Theme
key_theme_id (Primary Key)
Theme Keyword
access#

Keyword Theme Map
ii_id
key_theme_id

Keyword Place
key_place_id (Primary Key)
Place Keyword
access#

Keyword Place Map
ii_id
key_place_id

Keyword Stratum
key_stratum_id (Primary Key)
Stratum Keyword
access#

Keyword Stratum Map
ii_id
key_stratum_id

APPENDIX D

The MSOFT utility

Flowchart for the MSOFT utility

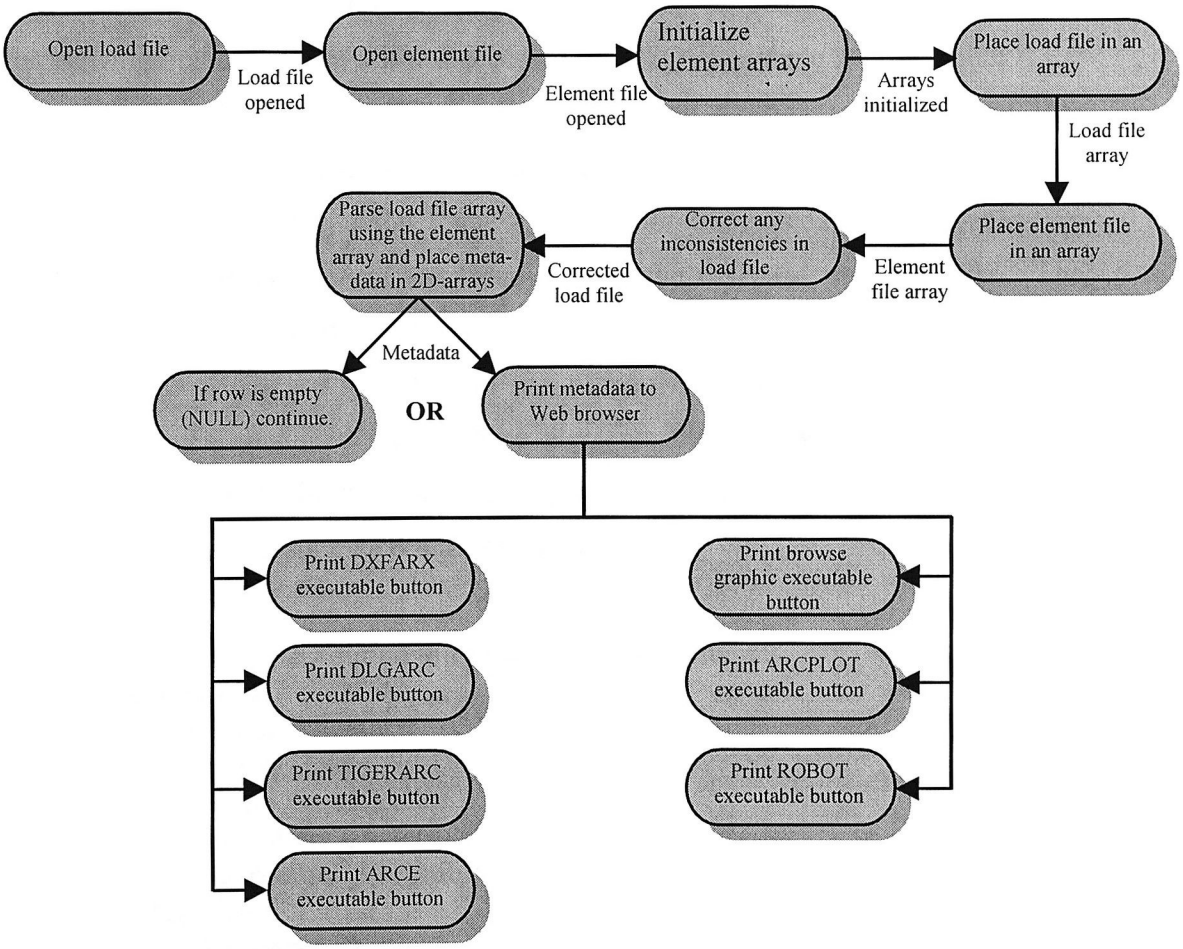


Figure 1 The MSOFT program that equates metadata directly to on-line software.

Functional description of the MSOFT utility

The MSOFT utility operates similar to the MLOADER parsing programs. In fact, the MSOFT utility is among the last GMS components develop by merging some of the code of the parsing programs and the GMS search-engine, which is described in chapter four. The utility scans a load file created by the MPARSER utility and examines the values of specified metadata elements (Figure 1). If the metadata file references one or more on-line geospatial data sets of a particular format, then HTML buttons are printed to the Web browser, which send command line arguments to the on-line conversion software, discussed in chapter 4.1. HTML buttons are also provided for on-line graphic files which when pressed, upload the file to the browser for viewing. Other on-line utilities accessed by MSOFT include a Web Robot if the metadata file references an HTML file, a coverage plotting utility and an E00 uncompress utility if the metadata references and Arc/Info interchange file.

The effectiveness of the MSOFT utility depends on the existence and organisation of certain elements and metadata values within the CSDGM metadata file. The program reads the following section and element names from the metadata file. Element names that are in bold text are of particular interest because it is the metadata values contained within these elements that are written to the HTML executable buttons as hidden fields and then subsequently sent to various on-line software programs as command line arguments.

Elements for transferring digital information over the Internet:

Format_Name,
Format_Version_Number,
Format_Version_Date,
Format_Version_Specification,
Format_Information_Content,
File_Decompression_Technique,
Transfer Size,
Network_Resource_Name,
Network_Address,
Access_Instructions,
Host_Operating_System,
Fees,
Odering_Instructions,
Turnaround_Time,

Citation related information for geospatial data:
Originator,
Title,

Information for on-line graphic files that pictorially represents part of the data set:
Browse_Graphic_File_Name,
Browse_Graphic_File_Description,
Browse_Graphic_File_Type,
Browse_Graphic_File_Online_Linkage.

The metadata above is accessed via a similar mechanism as used in the MLOADER utility. However, instead of entering the metadata into the metadatabase, it is either printed to the Web browser and/or inserted into HTML executable buttons. The construction of HTML executable buttons is described in greater detail in the second section of chapter 4. The GMS search-engine uses these buttons rather extensively to link all the various components of GMS together. The MSOFT program uses the information extracted from the metadata file to provide linkages to relevant conversion utilities on the server. The element Format_Name is used to determine the appropriate data-conversion utilities to use with the on-line geospatial data set, which is referenced by the Network_Address element as a URL. Figure 2 shows the C code used to accomplish this:

Directing system output to the Web browser:

```
if (strcasen(bg_file_name, "NULL", 4) && strcasen(bg_file_on-line_link, "NULL", 4))
{ HTML executable button if the graphic file name and URL address are NOT NULL }
if (!strcasen(format, "DXF", 3))
{
 HTML executable button for DXFARC conversion utility
  printf("\n<form ACTION=\"URL address of CGI script\" METHOD=POST>");
.
.
.
  printf("</form>");
}
if (!strcasen(format, "DLG", 3)) { HTML executable button for DLGARC conversion utility }
if (!strcasen(format, "TGRLN", 5)) { HTML executable button for TIGERARC conversion utility }
if (!strcasen(format, "ARCE", 4))
{ HTML executable button E00 uncompress and ARCPLOT utilities }
if (!strcasen(URL_address, "ROBOT", 5)) { HTML executable button for the Web robot }
```

Figure 2 C code that constructs the HTML executable buttons for the MSOFT utility.

In compliance with the CSDGM standard, DXF, DLG and TGRLN in the column *Format_Name* are standard acronyms indicating the URL in the column

Network_Address references data in those proprietary data formats (FGDC, 1994). The E00 export file format is represented by the ARCE designation, which is used to activate the E00 uncompress and ARCPLOT utilities. If the element *Network_Address* contains a URL address that terminates with the HTML file designation, then the MSOFT program provides a button to activate the Web robot.

APPENDIX E

A CSDGM metadata file and MPARSER output

!*!*!*

West Bounding Coordinate: -108.5000|
East Bounding Coordinate: -108.3750|
North Bounding Coordinate: 40.0000|
South Bounding Coordinate: 39.8750|

Keywords
Theme
Theme Keyword Thesaurus: None|
Theme Keyword: Mule deer|
Theme Keyword: Wildlife|
Theme Keyword: Winter range|
Theme Keyword: Summer range|
Place
Place Keyword Thesaurus: None
Place Keyword: Colorado|

Access Constraints: Public|
Use Constraints: None|

Data Quality Information

Attribute Accuracy
Attribute Accuracy Report: Created a check plot which was then visually verified by the wildlife biologist in a reinterative process until all errors are corrected and the dataset is accepted by the biologist.|

Logical Consistency Report: Polygons are closed and adjacent polygons do not have identical attributes.|

Completeness Report: Mule deer range for all seasons has been completely digitized for this quad.|

Positional Accuracy
Horizontal Positional Accuracy
Horizontal Positional Accuracy Report: Mule deer habitat compiled based on field observations by BLMCSO and Colorado Division of Wildlife (CDOW) wildlife biologists. Compiled data are transferred to Mylar overlays on top of USGS 7.5 minute quads.|

Lineage:
Source Information
Source Citation
Originator: CDOW, Grand Junction District (CDOWGJD).|
Publication Date: 08-Feb-1979.|
Title: Mule deer habitat observations.|
Geospatial Data Presentation Form: remote-sensing image|
Source Scale Denominator: 24000.|
Type of Source Media: Photo|
Source Time Period of Content
Range of Dates/Times
Beginning Date: 01-Apr-1977|
Ending Date: 30-Mar-1978|
Source Currentness Reference: Ground condition.|
Source Citation Abbreviation: CDOWGJD|
Source Contribution: Photographs which had polygons drawn on them to represent mule deer seasonal habitats.|

Source Citation
Originator: BLM, White River Resource Area (BLMWRRA)|
Publication Date: 08-Feb-1979|
Title: Mule deer habitat observations.|
Geospatial Data Presentation Form: Map|
Source Scale Denominator: 24000.|

Type of Source Media: Stable base mylar.|
Source Time Period of Content
Range of Dates/Times
Beginning Date: 01-Apr-1977|
Ending Date: 30-Mar-1978|
Source Currentness Reference: Ground condition.|
Source Citation Abbreviation: BLMWRRA|

Source Contribution: Transferred polygons from the photos provided by CDOWGJD to stable base material over a USGS 7.5' topo map. Biologist modified labels to conform to BLM standards. GIS Specialist converted to digital model.|

Process Step:
Process Description: Seasonal flights were made over the area and photographs taken. Wildlife specialists identified herds of deer on the photos and extrapolated ranges.
Source Used Citation Abbreviation: CDOWGJD
Process Date: 8-Mar-79.
Source Produced Citation Abbreviation: NA
Process Contact
Contact Organization Primary
Contact Organization: Colorado Division of Wildlife
Contact Address
Address Type: Physical address.
City: Grand Junction
State: Co
Process Step:
Process Description: ADS Capture: Digitize polygons and attributes; ADS Edit: Edgefit (default tolerances), Closure (default tolerances), Polygon.
Source Used Citation Abbreviation: BLMWRRA.
Process Date: 8-Mar-79.
Source Produced Citation Abbreviation:
Process Contact
Contact Organization Primary
Contact Organization: BLM, White River Resource Area.
Contact position: Wildlife biologist.
Contact Address
Address Type: Mailing adress
Address:73544 Highway 64
City: Meeker
State or Province:Co.
Postal Code: 81641
Contact Voice Telephone: (303) 358-8939.

Spatial Data Organization Information:

Direct Spatial Reference Method: Vector (Polygon data).|

Spatial Reference Information

Horizontal Coordinate System Definition
Planar
Map Projection
Map Projection Name: Lambert Conformal Conic|
Map Projection Parameters
Lambert Conformal Conic
Standard Parallel: 37, 42|
Longitude of Central Meridian: -109|
Latitude of Projection Origin: 36|
False Easting: 5000000|
False Northing: 0|

Planar Coordinate Information
Planar Coordinate Encoding Method: Coordinate pair|
Coordinate Representation
Abscissa Resolution: .61|
Ordinate Resolution: .61|
Planar Distance Units: meters|
Geodetic Model
Horizontal Datum Name: North American Datum of 1927|
Ellipsoid Name: Clarke 1866|
Semi-Major Axis: 6378206.4|
Denominator of Flattening Ratio: 294.98|

Entity and Attribute Information

Overview Description
Entity and Attribute Overview: Mule Deer Range consists of polygons used to identify seasonal range habitat for Mule deer. Polygon attributes describe the season that the mule deer are found on the land represented by the polygon.|
Entity and Attribute Detail Citation: The attribute coding standards can be found listed in the Standard Theme Specifications Users Guide available through the Colorado State Office.|

Distribution Information:

Distributor

Contact Organization Primary
Contact Organization: BLM, White River Resource Area.|
Contact Position: Wildlife biologist.|
Contact Address
Address Type: Mailing address.|
Address: 73544 Highway 64|
City: Meeker|
State: Co.|
Postal Code: 81641|
Contact Voice Telephone: (303) 358-8939.|

Distribution Liability: No warranty is made by the Bureau of Land Management for use of this data for purposes not intended by BLM. These maps are not legal documents and not intended to be used as such.|

Standard Order Process

Digital Form
Digital Transfer Information
Format Name: Automated Digitizing System (ADS)|
Digital Transfer Option
Offline Option
Offline Media: Prime 9-track|
Recording Capacity
Recording Density: 800, 1600, 6250|
Recording density units: characters per inch|
Recording Format: Primos MAGSAV, MAGNET|
Fees: MAGSAV - Cost of tape (10.00) + processing fee (50.00) + mailing fee (5.00) = total fees (65.00)
MAGNET - Additional processing fees|

Metadata Reference Information:

Metadata Date: 14-OCT-1994|
Metadata Contact
Contact Organization Primary
Contact Organization: BLM, White River Resource Area.|
Contact Position: Wildlife biologist.|
Contact Address
Address Type: Mailing address.|
Address: 73544 Highway 64|
City: Meeker|
State: Co.|
Postal Code: 81641|
Contact Voice Telephone: (303) 358-8939.|
Metadata Standard Name: Content Standards for Digital Geospatial Metadata|
Metadata Standard Version: 19940608|
??

!*!*!*

****
Publication Place
Identification Information
1
Unpublished
****
Abstract
Identification Information
1
Polygons digitized, using the BLM's Automated Digitizing System, from data transferred onto USGS quads. Each polygon is labeled with the season during which the mule deer occupy the area represented by the polygon.
****
Purpose
Identification Information
1
To represent digitally the mule deer ranges that exist on this 7.5 minute quad. Data captured for the Piceance Resource Management Plan (PRMP).
****
Currentness Reference
Identification Information
1
Ground condition
****
Progress
Identification Information
1
Complete
****
Access Constraints
Identification Information
1
Public
****
Use Constraints
Identification Information
1
None
****
West Bounding Coordinate
Identification Information
1
-108.5000
****
East Bounding Coordinate
Identification Information
1
-108.3750
****
North Bounding Coordinate
Identification Information
1
40.0000
****
South Bounding Coordinate
Identification Information
1
39.8750

****
Logical Consistency Report
Identification Information
1
Polygons are closed and adjacent polygons do not have identical attributes.
****
Completeness Report
Identification Information
1
Mule deer range for all seasons has been completely digitized for this quad.
****
Beginning Date
Identification Information
1
01-JAN-1977
****
Ending Date
Identification Information
1
01-JAN-1978
****
Theme Keyword
Identification Information
1
Mule deer
****
Theme Keyword
Identification Information
2
Wildlife
****
Theme Keyword
Identification Information
3
Winter range
****
Theme Keyword
Identification Information
4
Summer range
****
Place Keyword
Identification Information
1
Colorado
****
Originator
Lineage
2
CDOW, Grand Junction District (CDOWGJD).
****
Originator
Lineage
3
BLM, White River Resource Area (BLMWRRA)
****
Publication Date
Lineage

2
08-Feb-1979.
****
Publication Date
Lineage
3
08-Feb-1979
****
Title
Lineage
2
Mule deer habitat observations.
****
Title
Lineage
3
Mule deer habitat observations.
****
Geospatial Data Presentation Form
Lineage
1
remote-sensing image
****
Geospatial Data Presentation Form
Lineage
2
Map
****
Source Scale Denominator
Lineage
1
24000.
****
Source Scale Denominator
Lineage
2
24000.
****
Type of Source Media
Lineage
1
Photo
****
Type of Source Media
Lineage
2
Stable base mylar.
****
Source Currentness Reference
Lineage
1
Ground condition.
****
Source Currentness Reference
Lineage
2
Ground condition.
****

Source Citation Abbreviation
Lineage
1
CDOWGJD
****
Source Citation Abbreviation
Lineage
2
BLMWRRA
****
Source Contribution
Lineage
1
Photographs which had polygons drawn on them to represent mule deer seasonal habitats.
****
Source Contribution
Lineage
2
Transferred polygons from the photos provided by CDOWGJD to stable base material over a USGS 7.5' topo map. Biologist modified labels to conform to BLM standards. GIS Specialist converted to digital model.
****
Beginning Date
Lineage
2
01-Apr-1977
****
Beginning Date
Lineage
3
01-Apr-1977
****
Ending Date
Lineage
2
30-Mar-1978
****
Ending Date
Lineage
3
30-Mar-1978
****
Direct Spatial Reference Method
Spatial Data Organization Information
1
Vector (Polygon data).
****
Map Projection Name
Spatial Data Organization Information
1
Lambert Conformal Conic
****
Entity And Attribute Overview
Spatial Data Organization Information
1
Mule Deer Range consists of polygons used to identify seasonal range habitat for Mule deer. Polygon attributes describe the season that the mule deer are found on the land represented by the polygon.
****

Entity and Attribute Detail Citation
Spatial Data Organization Information
1
The attribute coding standards can be found listed in the Standard Theme Specifications Users Guide available through the Colorado State Office.
****
Distribution Liability
Distribution Information
1
No warranty is made by the Bureau of Land Management for use of this data for purposes not intended by BLM. These maps are not legal documents and not intended to be used as such.
****
Format Name
Distribution Information
1
Automated Digitizing System (ADS)
****
Fees
Distribution Information
1
MAGSAV - Cost of tape (10.00) + processing fee (50.00) + mailing fee (5.00) = total fees (65.00)
MAGNET - Additional processing fees
****
Contact Organization
Distribution Information
1
BLM, White River Resource Area.
****
Contact Position
Distribution Information
1
Wildlife biologist.
****
Address Type
Distribution Information
1
Mailing address.
****
Address
Distribution Information
1
73544 Highway 64
****
City
Distribution Information
1
Meeker
****
Postal Code
Distribution Information
1
81641
****
Contact Voice Telephone
Distribution Information
1
(303) 358-8939.
****

Metadata Date
Metadata Reference Information
1
14-OCT-1994
****
Metadata Standard Name
Metadata Reference Information
1
Content Standards for Digital Geospatial Metadata
****
Metadata Standard Version
Metadata Reference Information
1
19940608
****
Contact Organization
Metadata Reference Information
2
BLM, White River Resource Area.
****
Contact Position
Metadata Reference Information
2
Wildlife biologist.
****
Address Type
Metadata Reference Information
2
Mailing address.
****
Address
Metadata Reference Information
2
73544 Highway 64
****
City
Metadata Reference Information
2
Meeker
****
Postal Code
Metadata Reference Information
2
81641
****
Contact Voice Telephone
Metadata Reference Information
2
(303) 358-8939.
****

!*!*!*

APPENDIX F

The HTTP and FTP upload utility

This appendix provides a short description of the HTTP and FTP upload programs, which constitutes perhaps one of the most important GMS components. The HTTP upload program initially parses a URL address using a C program called HTTP_PARSE. This program extracts the file name of the on-line data set from the address and assigns it to a variable within the CGI script. File names are truncated to thirteen characters to conform to the maximum allowable length the Arc/Info GIS allows for coverage names. The cache directory for GMS is then searched for data sets that match the file name extracted from the URL address. The flow diagram shown in Figure 1 shows the entire process.

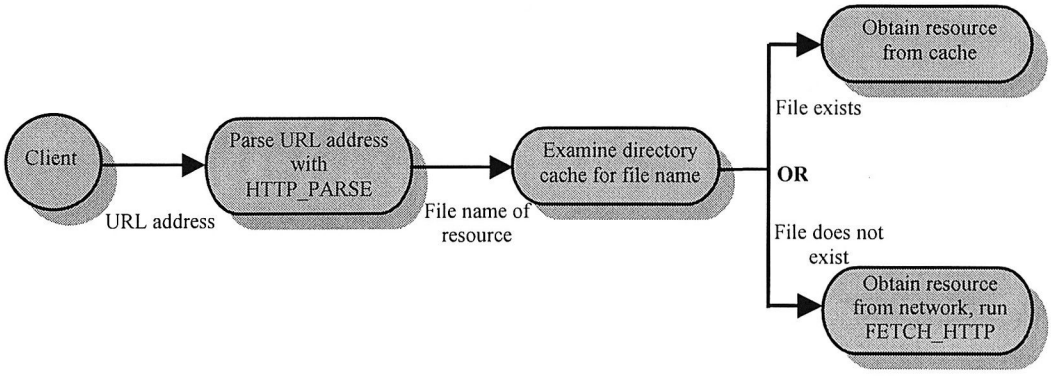


Figure 1 Flow diagram for the HTTP uploading program.

The most commonly used data sets are placed within the cache directory. If a match between the file name extracted from the URL address and one of the files within the cache is found, the system simply copies the data set from the cache to the directory space of the current process. Considering the large file size of many geospatial data sets, a directory cache is a simple method of saving processing time and bandwidth by freeing GMS from retrieving the resource from the network.

Alternatively, if the file name does not match any of the files in the cache, the data set is retrieved from the network using the FETCH_HTTP (Jones, 1995).

The FTP uploading program operates somewhat differently than its HTTP counterpart. The URL address is sent to one of two FTP parsing programs (FTP1 and FTP2). The reason for developing two FTP parsing programs is explained in the next sub-section. The two parsing programs not only extract a resource name from the URL address but also assemble a small DCL CGI script that is written to a temporary file, which is used to operate the FTP client program. The FTP client program is unlike FETCH_HTTP because it is a separate piece of software designed to operate interactively with the user to send and receive files from remote FTP sites. The small DCL scripts created by the parsing programs allows the FTP client to be used automatically. The flow diagram shown in Figure 2 shows the entire process.

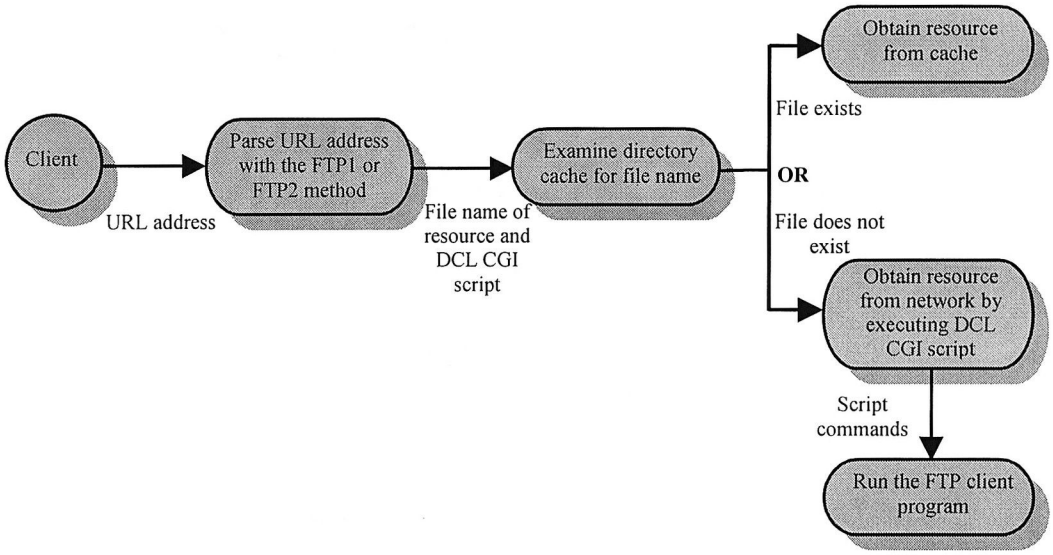


Figure 2 Flow diagram for the FTP upload program.

As in the case of the HTTP upload program, the file name extracted from the URL address is compared against the names of the files in the cache directory. If the file does not exist in the cache, then it is retrieved from the network by executing the DCL CGI script. The script starts the FTP client program that uploads the data set from a remote FTP site using the FTP 'get' command.

APPENDIX G

The geometric search subroutines

This section refers to the circle-line subroutine of Figure 1 and the line-line subroutine of Figure 2 used in the search-engine subroutines described in chapter 4. The two diagrams outline the exact sequence of tests in order to calculate the linear intersections between the circle/quadrilateral search areas and the boundaries of each database record.

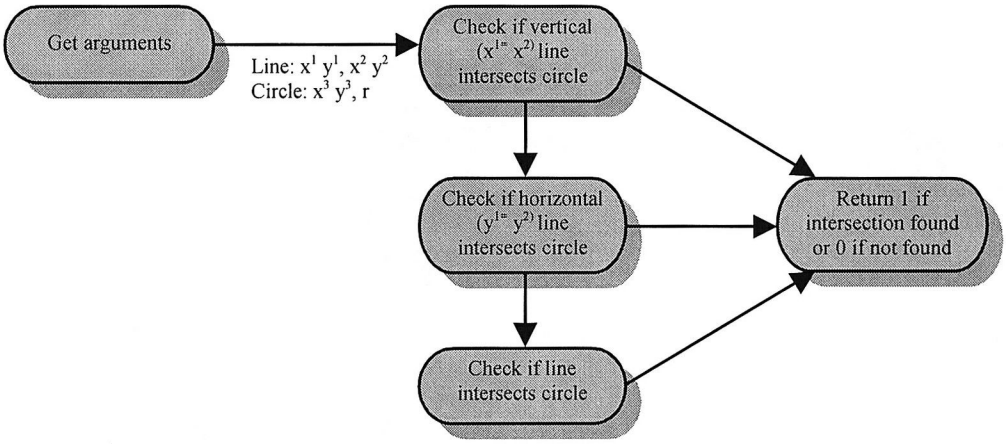


Figure 1 The circle-line subroutine that determines if a given line and circle intersect.

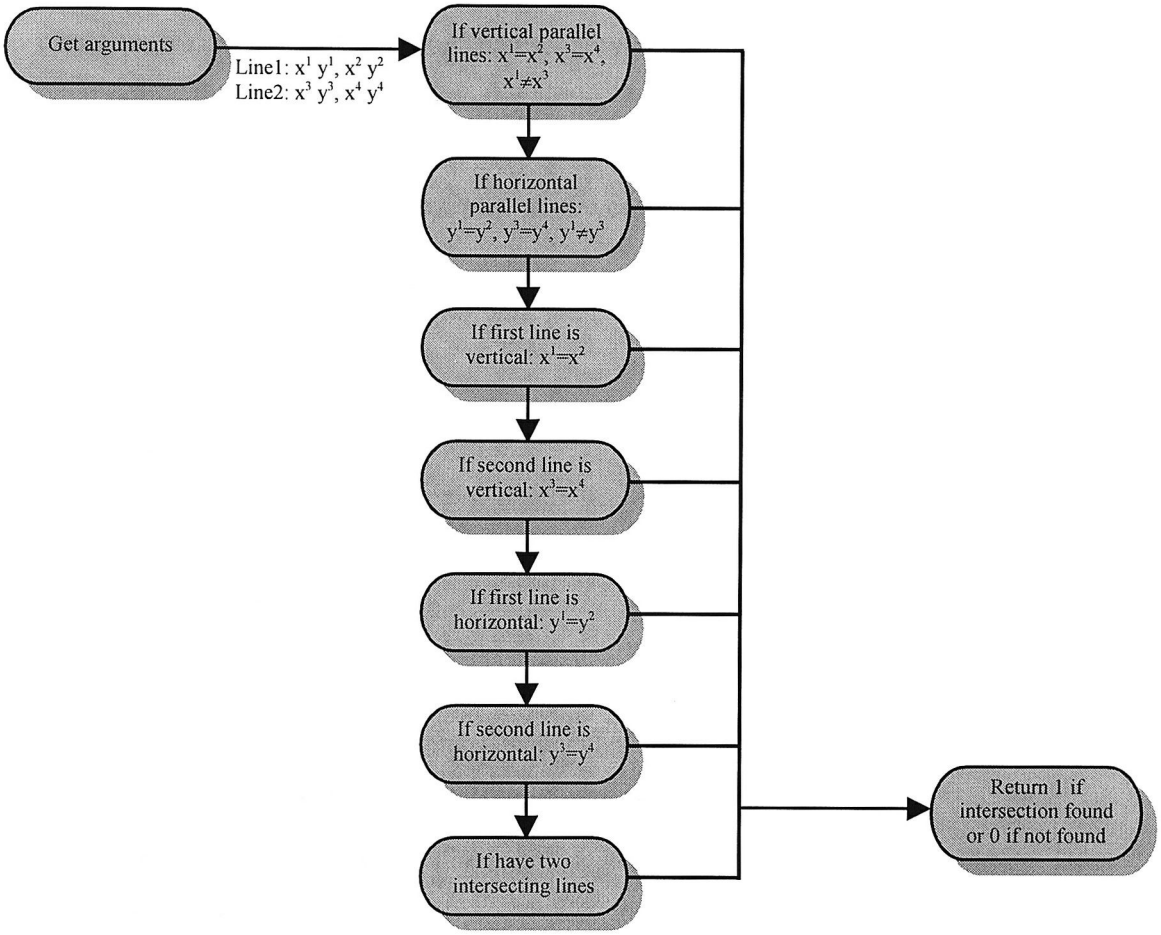


Figure 2 The line-line subroutine that determines if two given lines intersects.

The following appendix mathematically derives the equations used in both these subroutines, which determine intersection points between a circle and a line as well as between two separate lines.

The circle-line subroutine comprises of seven arguments, which include the x-y co-ordinates of the two extreme points that define a line, the x-y co-ordinate of the centre of the circle and the radius of the circle. There are three tests conducted using these arguments. Starting with the last test (Figure 1):

Given a line with *(x1, y1)* and *(x2, y2)* as the co-ordinates of its two points and a circle with *(x3, y3)* as the co-ordinates of its centre point and *r* as radius. Find at what points the line intersects with the circle.

Now, the equation of the line with the above co-ordinates is:

$$(y - y1) = \frac{(y2 - y1)}{(x2 - x1)}(x - x1)$$

$$y = \frac{(y2 - y1)}{(x2 - x1)}(x - x1) + y1$$

$$\frac{(y2 - y1)}{(x2 - x1)} = m$$

$$y = m(x - x1) + y1$$

$$y = mx - mx1 + y1$$

$$y1 - mx1 = n$$

$$y1 = mx1 + n$$

The equation of a circle with the above co-ordinates is:

$$(x - x3)^2 + (y - y3)^2 = r^2$$

$$(x - x3)^2 + (mx + n - y3)^2 = r^2$$

$$n - y3 = p$$

$$(x - x3)^2 + (mx + p)^2 = r^2$$

$$x^2 - 2x_3x + x_3{}^2 + m^2x^2 + 2mpx + p^2 = r^2$$

$$(1 + m^2)x^2 + (2mp - 2x_3)x + (p^2 + x_3{}^2 - r^2) = 0$$

$$(1 + m^2) = a$$

$$(2mp - 2x_3) = b$$

$$(p^2 + x_3{}^2 - r^2) = c$$

The two points that are the intersection between a circle and a line are:

$$Hence, ax^2 + bx + c = 0$$

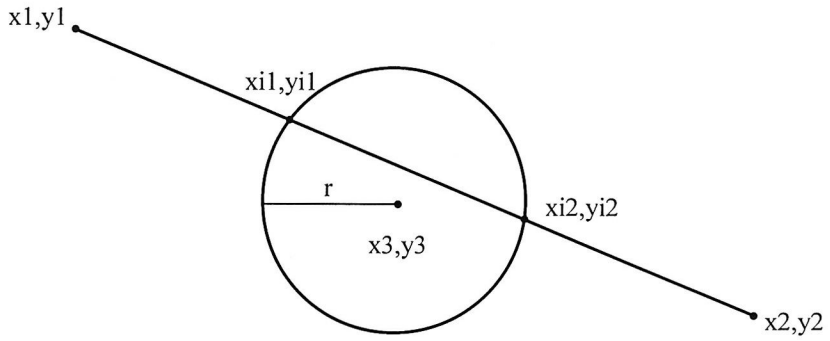
$$Now,$$

$$x_{i1} = \frac{-b + \sqrt{(b^2 - 4ac)}}{2a}$$

$$x_{i2} = \frac{-b - \sqrt{(b^2 - 4ac)}}{2a}$$

$$y_{i1} = mx_{i1} + n$$

$$y_{i2} = mx_{i2} + n$$



If the lines were horizontal, y^1 would equal y^2.

Given the Equation of a line is: $y = mx + c$

If the line is horizontal then $y1 = y2 = K_2$, where K_2 is a constant. The equation of the line then reduces to $y = K_2$.

The equation of a circle is: $(x - x3)^2 + (y - y3)^2 = r^2$

$$(x - x3)^2 + (K_2 - y3)^2 = r^2$$

$$(x - x3)^2 = r^2 - (K_2 - y3)^2$$

$$(x - x3) = \sqrt{r^2 - (K_2 - y3)^2}$$

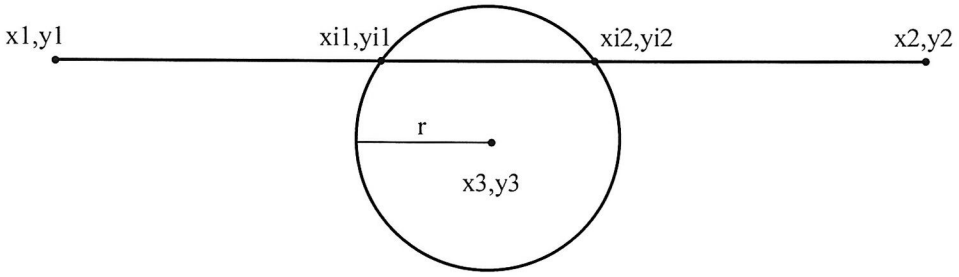
$$x_{i1} = +\sqrt{r^2 - (K_2 - y3)^2} + x3$$

or

$$x_{i2} = -\sqrt{r^2 - (K_2 - y3)^2} + x3$$

The intersection point between a horizontal line and a circle are, (x_{i1}, y) (x_{i2}, y).

Note, if $r^2 - (K_2 - y3)^2$ is a negative value then the line and the circle do not intersect.



If the lines were vertical, x^1 would equal x^2.

Equation of a line is: $y = mx + c$.

If the line is vertical then $x1 = x2 = K_1$, where K_1 is a constant. The equation of the line then reduces to $x = K_1$.

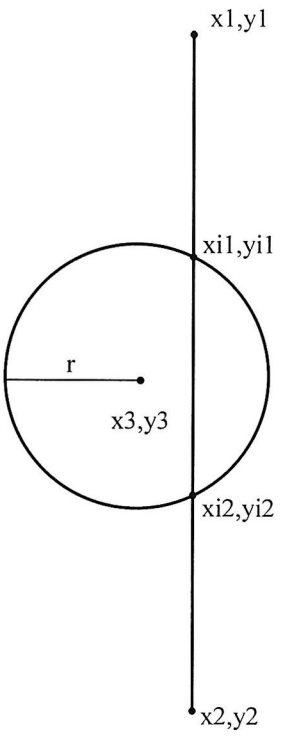
The equation of a circle is: $(x - x3)^2 + (y - y3)^2 = r^2$

$$(K_1 - x3)^2 + (y - y3)^2 = r^2$$
$$(y - y3)^2 = r^2 - (K_1 - x3)^2$$
$$(y - y3) = \sqrt{r^2 - (K_1 - x3)^2}$$
$$y_{i1} = +\sqrt{r^2 - (K_1 - x3)^2} + y3$$

or

$$y_{i2} = -\sqrt{r^2 - (K_1 - x3)^2} + y3$$

The intersection point between a vertical line and a circle are, (x, y_{i1}) (x, y_{i2}). Note, if $r2 - (K_1 - x3)^2$ is a negative value then the line and the circle do not intersect.



The line-line subroutine comprises of eight arguments, which include the four pairs of x-y co-ordinates that define the extreme points of each of the two lines. There are seven tests conducted using these arguments (Figure 2). The first two tests check if the lines are vertically or horizontally parallel. Given the first line with the following two points, (x1, y1) and (x2, y2) and the second line with the following two points, (x3, y3) and (x4, y4). Two vertical parallel lines would be defined as $x^1=x^2$, $x^3=x^4$, $x^1\neq x^3$ and two horizontal parallel lines would be defined as $y^1=y^2$, $y^3=y^4$, $y^1\neq y^3$.

The remaining test for the line-line subroutine determines intersections of two given lines if the first or second line is vertical, the first or second line is horizontal and finally if the two line are neither vertical nor horizontal. The equations of these tests are derived below.

Given two lines. The first line has *(x1, y1)* and *(x2, y2)* as the co-ordinates of its two points and the second line has *(x3, y3)* and *(x4, y4)* as the co-ordinates of its two points. The co-ordinates of the intersection point between these two lines are:

The equations of the two lines are:

$$y = m_1 x + n_1 \qquad y = m_2 x + n_2$$

$$m_1 = \frac{(y2 - y1)}{(x2 - x1)} \qquad m_2 = \frac{(y4 - y3)}{(x4 - x3)}$$

$$n_1 = y1 = m_1 x1 \qquad n_2 = y3 - m_2 x3$$

Equating y and solving for x give the following:

$$m_1 x + n_1 = m_2 x + n_2$$

$$m_1 x - m_2 x = n_2 - n_1$$

$$(m_1 - m_2)x = n_2 - n_1$$

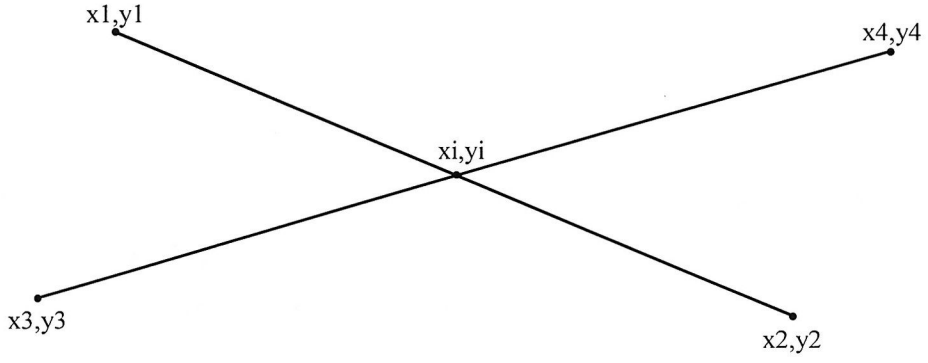
$$x = \frac{(n_2 - n_1)}{(m_1 - m_2)}$$

And y can be determined by substituting in either of the equations:

$$y = m_1 x1 + n_1$$

or

$$y = m_2 x3 + n_2$$



Given two lines. The first line has *(x1, y1)* and *(x2, y2)* as the co-ordinates of its two points is vertical. The second line has *(x3, y3)* and *(x4, y4)* as the co-ordinates of its two points. Find the co-ordinates of the intersection point between these two lines.

The equations of the two lines are:

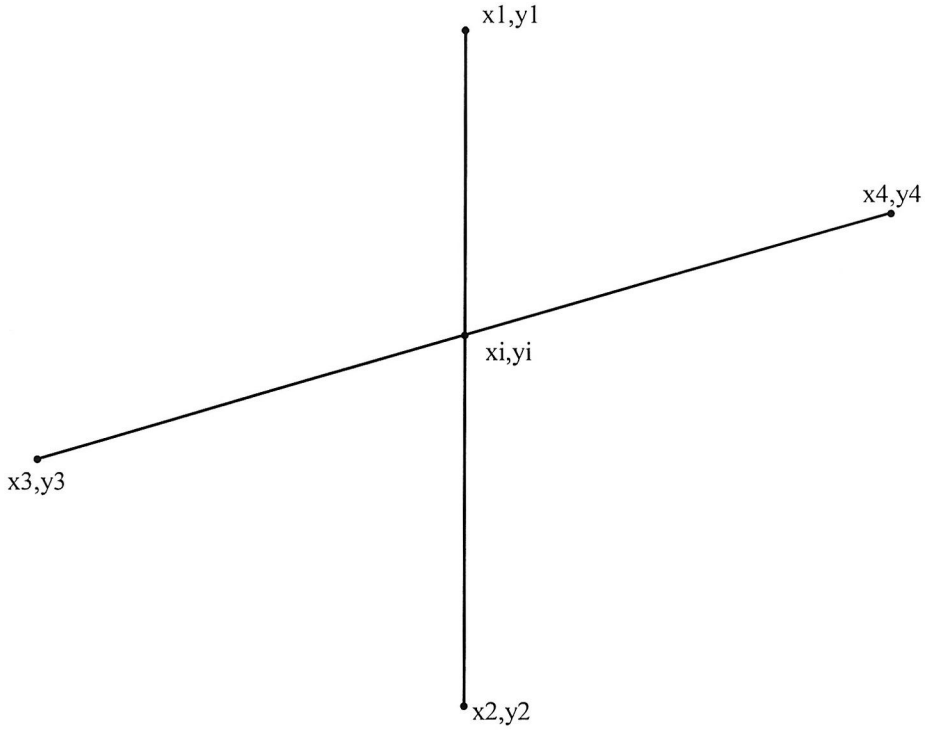
$$y = m_1 x + n_1 \qquad y = m_2 x + n_2$$

$$\because x1 = x2 = K_1 \qquad m_2 = \frac{(y4 - y3)}{(x4 - x3)}$$

$$x = K_1 \qquad n_2 = y3 - m_2 x3$$

Substituting K_1 into the equation of the second line: $y = m_2 K_1 + n_2$

The intersection point is: (K_1, y)



Given two lines. The first line has *(x1, y1)* and *(x2, y2)* as the co-ordinates of its two points is horizontal. The second line has *(x3, y3)* and *(x4, y4)* as the co-ordinates of its two points. Find the co-ordinates of the intersection point between these two lines.

The equations of the two lines are:

$$y = m_1 x + n_1$$
$$\because y1 = y2 = K_2$$
$$y = K_2$$

$$y = m_2 x + n_2$$
$$m_2 = \frac{(y4 - y3)}{(x4 - x3)}$$
$$n_2 = y3 - m_2 x3$$

Substituting K_2 into the equation of the second line:

$$K_2 = m_2 x + n_2$$

$$x = \frac{K_2 - n_2}{m_2}$$

The intersection point is: *(x, K_2)*

