DESIGN OF APPLICATION MODEL FOR GEOSPATIAL DATA CLEARINGHOUSE

Chan, Ho-chiu Elton

Thesis submitted in partial fulfillment of requirements

for the degree of Master of Philosophy in the Department

of Land Surveying & Geo-Informatics

Department of Land Surveying & Geo-Informatics

The Hong Kong Polytechnic University

2000

ABSTRACT

Geospatial applications consist of procedures applied against a number of data sets. The task of developing geospatial applications is usually time-consuming and working in trial-and-error basis. Meanwhile, archived applications are difficult to be reused. From the GIS users viewpoint, they need to understand what it was created, what was the background logic, what data was involved in the analysis and so on. It can be made easier, much efficient and understandable if validated spatial analysis procedures and modeling solutions can be managed in a systematical way that the geospatial applications can be shared and reused. This study develops a geospatial application model for providing a logical and systematical way to describe geospatial application. The model consists of three components including application component, implementation component and spatial data component, which are organized in a tree hierarchy.

To demonstrate the idea for the study, a documentation system was developed based on the application model with latest technologies such as Extensible Markup Language (XML), Document Object Model (DOM) and Java programming language.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACKNOWLEDGMENTS

I have to express my deep appreciation to my supervisor, Dr. LEE Yuk-cheung, who leads me to understand the basic concept, implementation and application of Geographic Information System technology. He also guided me how to handle the difficulties faced and gave me constructive suggestions in this research. I enjoyed very much working with him and have learned a lot in different aspects more than the technology. It is a pleasure to express my sincere appreciation for his support and guidance throughout my time.

In addition, I would like to acknowledge Mr. KIRK, P.A. for his fruitful suggestions and comments on my thesis. I would also like to thank the Department of Land Surveying and Geo-Informatics for their support throughout my time at the Hong Kong Polytechnic University. I wish also to thank my fellow graduate students, research assistants and technicians for providing excellent working environment.

Finally, I would like to thank my family and friends for giving me tireless personal supports, and to Berlina who gives me kindness helps and supports all the time.

Ho-chiu Elton, Chan
June 1999

This research was supported by the Central Research Grant under project number G.V.299 of The Hong Kong Polytechnic University.

This support is gratefully acknowledged.

CHAPTER 1
INTRODUCTION

1.1 Metadata

With the rapid development of Geographic Information System (GIS) in recent years, geospatial data development becomes a non-stop process that generates abundant information for different users in various communities. Although there is data available, GIS users usually do not understand them well. This becomes a major obstacle for performing spatial analysis. In fact, without successfully identifying an appropriate set of data, users might develop new data that is possibly already available in the community. This is a waste of time and definitely duplication of effort.

Furthermore, even if users can find a data set and they think it is useful for their application. They might not understand its important characteristics such as attributes contained, spatial reference used, degree of quality and so forth. Further analysis using this data may lead to unreliable products and misleading results. As a result, even data producers cannot ensure the quality of their products. That situation is so bad is the result of weak awareness of metadata and of mechanisms for metadata distributing. It is recognized that GIS users or analysts at different sites need to:

- Identify metadata within the data-sharing network,
- Browse the metadata,
- Understand the data via information such as data quality, usage and coordinate system,
- Determine the data fitness for the GIS application, and

- Access the data and perform analysis.

In the United States, 1994, under the Executive Order by the President [FGDC, 1994a], the Federal Geospatial Data Committee (FGDC) was established and started to promote and enforce geospatial data sharing through developments in geospatial metadata standards and related data entry utilities. The Content Standards for Digital Geospatial Metadata (CSDGM) was thus developed. Consequently, different kinds of metadata utility tools were developed, which help users to enter, validate and store metadata documents. For instance, metadata entry tools assist data producers to create metadata documents; metadata validation tools help users to find any syntax errors in the metadata document, while database management systems provide better storage and retrieval capabilities for managing metadata.

Nevertheless, the ultimate goal of creating metadata is to allow potential data users to browse and access the metadata in a networked environment. This requires the establishment of data clearinghouse for distributing metadata documents. In reality, it is a group of servers distributed in a network environment or the Internet. Each server would store particular set of metadata documents for users to browse, search and retrieve. The idea of making data to be distributed-stored is meant to minimize server overloading, and avoid single point failure problem [Peng and Nebert, 1997]. To ensure proper data searching, browsing and retrieving operations between servers, a communication protocol or mechanism is required.

At the time the FGDC designed the data clearinghouse concept, they adopted the Z39.50 as a basic protocol to search and retrieve metadata documents. The Z39.50 is an ANSI/ISO information retrieval service and protocol originally developed as a library standard for cataloguing bibliographic records. This is an application protocol, which simply defines a set of commands and messages to be sent and received. In this case, the set of commands and messages are designed for searching and retrieving information. In addition, the Z39.50 protocol relies on TCP/IP to ensure a communication between two machines can be established. TCP/IP is a protocol widely used in the Internet. The main reason to choose Z39.50 because it allows precise information searching at the time the data clearinghouse was developed. Another reason is that this protocol is well established and has been using by the library community for a long period of time. While development to metadata management in GIS is on going, the, Z39.50, TCP/IP, CSDGM and the data clearinghouse concept definitely play a significant role for building a model of metadata management in the community.

1.2 Existing Applications related to Geospatial Metadata

Analyzing the geospatial metadata management model, it helps to give a clear picture of what kind of application is related to geospatial metadata. In general, we can divide the model into two components. The first component, which serves as a basis of others, is a list of data items with fixed structure and comprehensive content used for describing geospatial data sets. It provides the guideline of what data item should be stored as metadata and forms the geospatial metadata standard.

On top of the standard is another component to serve metadata. It is merely a service providing a series of operating tools for creation, management and distribution of geospatial metadata. This service contains two sub-components: Metadata Management System and Metadata Information System. The primary goal of the

Metadata Management System is to provide an environment that enables convenient and efficient creation and storage of metadata. Applications of Metadata Management System include metadata entry, validation, maintenance, retrieval, indexing, and reporting. In other words, this is a database management system, which dedicates for managing metadata. A database management system is a system to provide an environment for retrieving and storing data in convenient and efficient manners [Korth and Silberschatz, 1991].

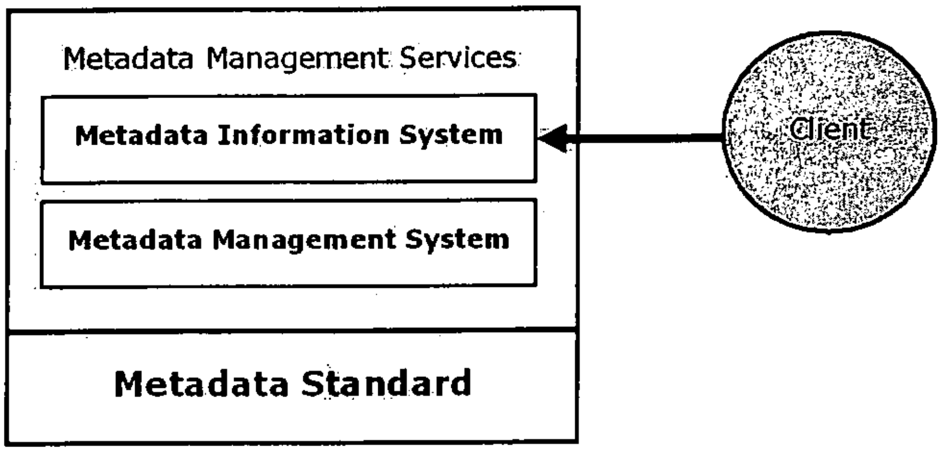

Metadata Management Services
Metadata Information System
Metadata Management System
Metadata Standard
Client

Figure 1.1 Geospatial metadata management model

In addition, an information system is a system for collecting, processing, storing, retrieving, and distributing information within an organization and between organizations in an internetworking environment [Bernus and Schmidt, 1998], while the Metadata Information System is kind of information system dedicate for managing metadata. This is composed of server and client. The server is a program that accepts request, and makes response to users who require metadata. The client, on the other hand, is a program for the users to submit requests to the server, as well as receives and displays the results. In this configuration, the Metadata Information System should also provide accurate searching capability to users. The following Figure 1.1 shows a diagram of the geospatial metadata management model.

The model is analogous to similar mechanisms or applications found in the current web technology. For example, with the help of web search engine, users can use client system such as Internet Explorer or Netscape to search the Internet. The operation simply requires users to submit a keyword as search parameter and results could be retrieved from a web database. Those results are hyperlinks, which in fact indicate the linked document contains the entered keyword. However, this mechanism is identical to free-text searches over a document that would return a large number of hyperlinks, no matter the linked contents is related or not. Therefore, this is not an effective and efficient mechanism to search information.

Metadata management differs from the present information discovery mechanism on the Internet in that when searching metadata, users expect the system to provide precise querying mechanism and to return meaningful search results. This is an essential requirement for establishing data fitness mechanism against GIS application, in fact, which is also, one of the main roles that the FGDC stated for the geospatial metadata.

The main role of metadata management service is to serve metadata. Based on this concept, we can develop applications that exploit metadata instead of typical geospatial data. One of the potential functions in the exploitation of metadata is to build application that can help users to find and access appropriate data sets for building particular geospatial applications.

1.3 The Problem

Although the applications mentioned in the previous section provide primary metadata handling mechanism, there is a deficiency. One of the major problems in the entire paradigm is that the searching mechanism is not well developed. This is highly affected by current technology employed to implement for the searching mechanism. Currently, the technology that we employ for searching and retrieving metadata is Z39.50. It solves a major problem that traditionally we use free text searching for document, which would come up with lots of unrelated search results. The Z39.50, on the other hand, provides mechanism for users to specify which keywords and keyword values to search. However, this is not a complete solution. There is a problem whenever users want to search data for an application, since users are required to provide keywords for the search. This may not be a serious problem for experienced GIS users, since this type of user is experienced in the selection of keyword to search. However, novice GIS users might encounter a typical question: which keyword is the right keyword to search? As a result, trial and error is the only approach to the problem.

From another point of view, whenever users want to search for data set, they should have an application in mind, i.e. spatial analysis what do they want to perform. However, it is common that users do not know the exact operations and required data to perform the application. In fact, the operations used are system specific and no two systems have operations using common terminology, parameters or command sequence. This gives a problem, especially for inexperienced GIS users who would require exploring much on different operations and parameters.

Even experienced users would face difficulty when they encounter little used operations. Typically intensive testing is required before actual use. We think that this work could be largely avoided, since it is possible that the unusual operation was used in the past. This problem might be due to the operations being too rare to recall. This is another common problem because we do not have a knowledge base to store and manage all the used operations or procedures. We would like to promote the reusability of application procedure, for the similar reasons to those for the development and exploitation of geospatial metadata.

For geospatial data, as we have mentioned, guidelines have been developed for describing data contents. This caters for metadata of the spatial data, which provides description of data. However, for geospatial applications, there is no existing guideline to help for describing and recording the knowledge of geospatial processing. This is the main problem in the current GIS environment and in fact this is the main problem we would like to explore in this project.

In this project, our motivation is to help users to determine appropriate geospatial resources, operations, and procedures for geospatial analysis. We propose to use an application model for solving the problem. Ultimately, we would like to make use of the application model and build geospatial application management system for users to document new application procedures, to retrieve, reuse and modify stored application procedures.

1.4 Project Objective

The main objective of this research project is to develop a geospatial application model using geospatial metadata.

An application model will be developed and it will provide a set of descriptions to record a geospatial application. In addition, the application model would be implemented as a software prototype to help users find appropriate geospatial data resources and to provide application operation with parameters. The benefit of the application model is to help users, especially those are not familiar with GIS data, to find, evaluate, and create new GIS data. This is also a major use of geospatial metadata i.e. to determine the fitness for use.

1.5 Project Summary

In this project, we will develop an application model, which defines the fundamental description of a geospatial application. The relation between the application model and the ordinary geospatial metadata is that the former is an application of the latter. According to the model, a document structure for describing geospatial applications was developed and guidelines given for users to create their own application document. Ultimately, a set of multiple application documents forms an application base. The main contribution of the application base is that it helps enable the production of geospatial knowledge and make it to be reusable and transferable, while it is convenient and efficient to be accessed. The prototype that we developed is based on an object-oriented design and analysis approach. It was written in JAVA with Java Foundation Classes (JFC). In addition, to build a test bed for communicating with geospatial data clearinghouse, we setup a metadata information system i.e. Isite on Linux, in which Linux is a Unix operating system. Meanwhile, a Z39.50-HTTP gateway with Apache web server, which is an HTTP server, was installed in the same location to enable Internet browser connecting to the metadata information server.

1.6 Organization of this Thesis

The remainder of this thesis describes the entire technology review, problem analysis, development and implementation of the application model in this research project in detail. The following is a brief description of remaining chapters:

- Chapter 2 provides a review of current implementation, development and functionality of geospatial metadata, metadata management systems, metadata information systems, as well as concepts and implementation of geospatial data clearinghouse. In addition, it will introduce the extensible markup language (XML) and its uses in the Internet.
- Chapter 3 describes the entire development process of the application model from problem analysis, system design and detail design.
- Chapter 4 discusses the implementation of the model and prototype that we developed.
- Chapter 5 provides a conclusion of the entire project, and possible further development in related areas.

CHAPTER 2
A REVIEW OF GEOSPATIAL METADATA MANAGEMENT

2.1 Geospatial Metadata

Metadata is data about stored data. More precisely, metadata can be described as the background information that describes content, quality, condition, history and other related characteristics of the data [Peng and Nebert, 1997].

Relational database management systems are designed to store data in table form. For user-defined or system tables, table names and corresponding columns will be stored in a cataloging table. Therefore, by querying this table, we can retrieve the name of existing tables and related information such as field names and data type. The cataloging table in this case stores metadata.

Different from ordinary databases, GIS databases store both geospatial and non-geospatial data. Data that describes the contents of this kind of database is called geospatial metadata. To manage geospatial metadata, it is necessary to document the geospatial component that is typically stored in a graphic file, as well as non-geospatial data that is commonly stored in ordinary databases. Thus, it is a good idea to specify a typical set of descriptive items. In addition, the set of descriptive items could be used to build a common language for exchanging metadata among the GIS community and this is a pre-emptive process to enable data sharing. This is also a basis for developing metadata standardization such as CSDGM. According to FGDC, the major uses of metadata are to:

- Maintain an organization's internal investment in geospatial data,

- Provide information about an organization's data catalogues, clearinghouses, and brokerages, and
- Provide information needed for processing and interpreting data to be received through a transfer from an external source.

Besides, according to FGDC, the geospatial metadata plays the following roles:

Availability – data needed to determine the sets of data that exist for a geographic location;

Fitness for use – data needed to determine if a set of data meets a specific need;

Access – data needed to acquire an identified set of data;

Transfer – data needed to process and use a set of data.

In other words, the building of geospatial metadata is to facilitate data usability, data sharing and investment protection on data creation. It also implies that geospatial metadata is an essential component for ensure data completeness. In this case, geospatial data is no longer a combination of geospatial data and attribute data, its metadata is a necessity to build a completed geospatial database.

2.1.1 Standards of Metadata

Without a unified guideline to create geospatial metadata, data sharing, determining fitness for use and data quality control is still not possible. Hence, several standards organizations are working on developing metadata standards. One of the best known metadata standards is the "Content Standards for Digital Geospatial Metadata" (CSDGM), which was developed by the FGDC in 1994. As well, the International Organization for Standardization TC211 [ISO, 1997] is also working on defining the ISO metadata standard based on the work of FGDC. Meanwhile, there are similar developments from the Australian and New Zealand Land Information Center [ANZLIC, 1997], and from the Comité Européen de Normalisation TC287 [CEN, 1997]. Among these efforts, the development of CSDGM and its supporting software are relatively more mature in term of accessibility and availability. In this study, we would use CSDGM as a basis for cataloguing geospatial metadata.

2.1.1.1 Content Standards for Digital Geospatial Metadata

Content Standards for Digital Geospatial Metadata (CSDGM) provides 311 elements, which are either mandatory or optional, and they are further grouped into seven main sections and three supporting sections for describing geospatial data. The following are the seven main sections, which help to describe different characteristics of geospatial data:

1. Identification Information.
2. Data Quality Information.
3. Spatial Data Organization Information.
4. Spatial Reference Information.
5. Entity and Attribute Information.
6. Distribution Information.

7. Metadata Reference Information.

In addition, the following are three supporting sections that are repeatable within the main sections and are not be used independently.

1. Citation Information.
2. Time Period Information.
3. Contact Information.

The main sections form a hierarchical structure starting from a root called "metadata". In Figure 2.1, the lines show relationships between the main sections and their corresponding supporting sections.

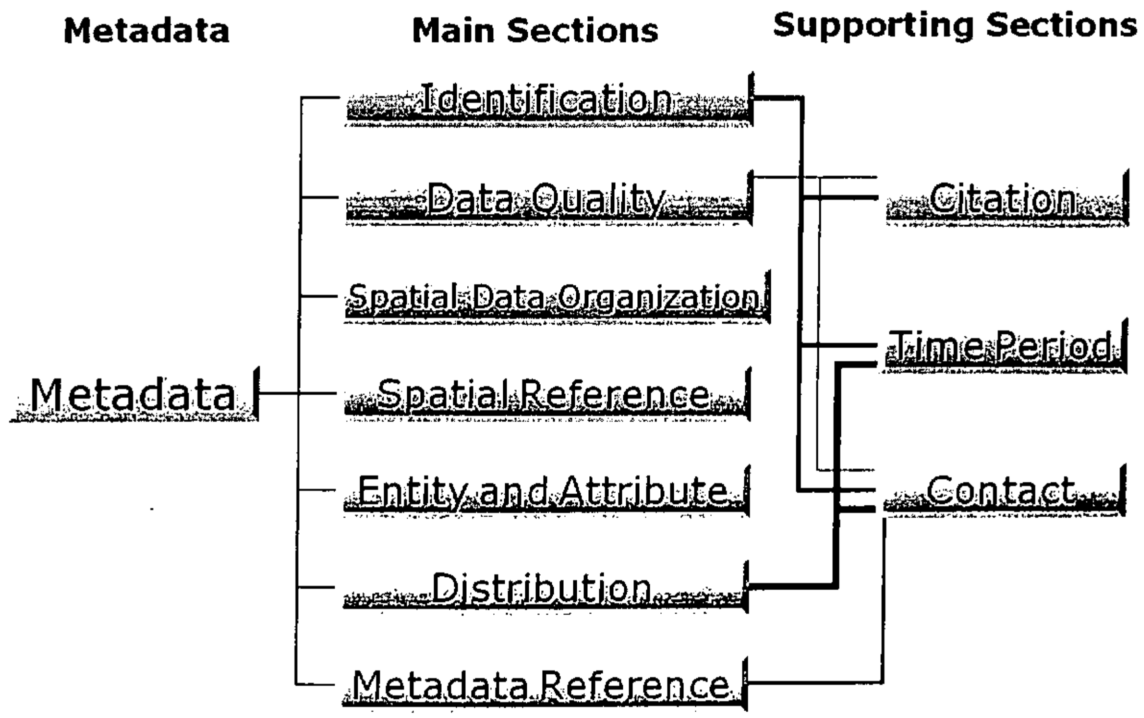


Figure 2.1 The hierarchical structure of CSDGM [FGDC, 1994b]

The metadata document is a hierarchical structure that can branch out to view its descendants under each section. There are compound elements, e.g. Element 1.4 Status, and/or data elements, e.g. Element 1.4.1 Progress. Compound element is a collection of data element(s) and/or other compound element(s). A data element on the other hand is a primitive item that cannot be subdivided, and a value will be assigned to it. In order to help control data input, a domain, containing a list of valid values is fully recorded in the standard that is used to assist assigning value to a data element.

For instance, Section 1: "Identification Information" contains both compound elements and data elements. Two of the compound elements are Element 1.4 "Status" and Element 1.7 "Access Constraints". The compound element 1.4 "Status" describes the state of and maintenance information about the data set, which contains two data elements 1.4.1 "Progress" and 1.4.2 "Maintenance and Update Frequency". These two data elements are primitive. The metadata standard provides a list of possible values for selection. For example, element "Progress" contains three domain values: Complete, Incomplete and Planned. All these are the basic structure of a metadata document.

In addition, the standard contains some administrative information such as liability statement, contact information, and distribution information that are generally not present in GIS databases. Nevertheless, the data producer should provide the information accordingly.

2.1.2 Level of Granularity

Although the metadata standard provides structure and list of keywords for describing data sets, deciding on a level of granularity is practically a common concern when documenting geospatial metadata. In reality, geospatial data is not always developed from consistent sources. There are cases that data might be derived from heterogeneous data sources for the whole series or for particular tiles within a single series. Some of the metadata contents in this case might vary in a high degree of inconsistency. If users want to document all the differences within the series, this would generate a very long documentation.

From our practical experience in preparing metadata document based on CSDGM, the followings are considerations for deciding the level of granularity:

- Consider the information that you expect to deliver through the metadata document and the application that might require it.
- Avoid including useless or ambiguous information into the metadata.
- Sometimes, a series of data set is quite similar to each other. In that case, it might be possible to produce one metadata document for the entire series and describes their differences in appropriate places such as Element 1.2.1 "Abstract" and Element 1.2.3 "Supplemental Information".
- If it is necessary to produce a separate metadata document for each of the data sets in a series, another document could be produced for the entire series. The current CSDGM does not support hierarchy of metadata documents. However, Element 1.14 "Cross Reference" and 1.2.3 "Supplemental Information" could be "borrowed" for this purpose.

2.1.3 Documenting Data Quality

Within CSDGM, the most difficult part for documentation is the data quality section of geospatial data. This section is composed of five compound elements: Attribute Accuracy, Logical Consistency, Completeness, Positional Accuracy, and Lineage.

The first element is "Attribute Accuracy", which in turn contains two compound elements: "Attribute Accuracy Report" and "Quantitative Attribute Accuracy Assessment'. The former records any testing methodology and result of accuracy assessment that was applied to the data set. The latter, an optional element, is another presentation method on attribute accuracy that yields a quantitative value. The second element is "Logical Consistency Report", which records the tests on topological consistency and attribute consistency of the data set. Regarding topological tests on geospatial data, it should detect the following problems:

1. Existence of duplicated lines;
2. Existence of overshoots or undershoots;
3. Existence of very small polygons or lines very close to each other;
4. Existence of line intersections.

With regard to attribute test, it should able to detect any invalid attribute against its domain value as well as any wrong input such as text label against the master source. The third element is "Completeness Report", which records any omission, selection criteria, and generalization applied to derive the data set. The fourth element is "Positional Accuracy', which uses an organization similar to the "Attribute Accuracy" element. It contains two compound elements, "Horizontal

Positional Accuracy" and "Vertical Positional Accuracy". These elements are used to record an estimation of the positional accuracy of the data set. Similar to element 2.1 "Attribute Accuracy", there is a quantitative assessment which provides an alternative method to recording positional accuracy. The last part of the data quality section is "Lineage", which records the history of the data set creation that might include the process, parameters, source data and information about parties involved. The content of this element is not rigid and allows users to enter appropriate information as preferred.

Among all these elements, the logical report, the completeness report and the lineage can easily be determined, since the information is generated and known during typical data production process. For instance, before doing anything, users should know if the potential data set is a subset of a larger data set, which is an answer of the data completeness. Hence, information on how to derive the data set is also known. Next, it is a must to build and maintain topological information among the data set before doing any topological analysis. The correctness of the attribute value can be determined against a valid domain list. However, it is difficult to determine the correctness about attribute value against data sources especially if the data size is large. Sampling methods can be employed to determine this. Users normally ignore this procedure, since it is a cumbersome, time-consuming task. It is a norm that there is no liability to check correctness of data. The same situation occurs in determining the positional accuracy, this is a complicated question since positional error is difficult to be modeled or visualized which involves various steps in the entire data production process.

To document the attribute and positional accuracy, CSDGM provides two approaches: quantitative method and qualitative method. The quantitative method uses an absolute value to reflect the correctness of the data set in both attribute and positional accuracy. To use this method, users should state a methodology that can be applied to the data set and yields a value to represent the correctness. However, it is criticized that if the selected method capable to reflect the error condition. On the other hand, the qualitative method uses some deductive descriptions to reflect the condition of the data set. Sometimes, relative value between the data set and its source can be used. Users might also base on their experience to state the data set condition. For instance, the following statement is used as a qualitative measure of positional accuracy in one of our metadata creation projects in preparing geospatial metadata:

"The horizontal positional accuracy of the field data is about 20 meters. This value was estimated by Geotechnical Engineers from their field experience. Accuracy of the digital data is affected by its source documents. With regard to the Geology Digital Map, the accuracy of the source document is compliant with the HGM20 specification." [Lee and Chan, 1998]

As mentioned, data producers are not aware of the importance for determining data quality. With the promotion and use of metadata, it should, on the other hand, becomes the data producers' responsibility to provide information on how good their data can be provided.

The standard of metadata and its creation is a foundation of the entire geospatial metadata management paradigm. This foundation enables users to communicate, exchange and prepare the geospatial metadata properly with full understanding of the purpose, contents and usage of geospatial metadata. To continue the discussion, in the following sections, we start to explore the management portion of the paradigm.

2.2 Metadata Management System

Metadata management system in general is a database system used to provide effective and efficient management of metadata. Employing the geospatial metadata as basis, the role of metadata management system is to provide a persistence metadata store accompany with operations such as data entry, data updating and data retrieval. In fact, there are a number of metadata management systems available in the GIS community. Most of them are publicly available as freeware. Among these management systems, they can be classified into two different persistence store technologies, which include ordinary file system and Relational Database Management System (RDBMS). These persistence stores help keeping the existence of stored data beyond the executable time of program, which processes the data. In general, the data would be written in physical memory such as hard drive for anytime system start, which can be retrieved and processed again.

2.2.1 Using File System for Storage

It is common to use a file system to store data, because it is generally included in an operating system. Meanwhile, file system is simple and easy to implement with no additional cost. However, file system itself has no capability to store semantic and manipulate data. Within a file, it is just a sequence of bytes stored and is treated as a single unit in the system. It can only provide limited information about the data set such as filename, lasted modified date and the file size. The file structure employed in a file system is dedicated for storing data. For example, beside the file contents, there is a file header, which tells the system about the file type and file attribute such as if it is read-only, hidden or system, and remaining is the file content. A programming language is necessary to provide richer data structure and related data maintenance utilities in handling users' data. Expected data management services like indexing and querying is insufficient or even not available. It is possible to develop sophisticated program to manipulate data files. However, this is a form of development on database system and is unlikely to be cost effective. Regarding the current implementation that uses file system for storing metadata, it can further be classified into two categories based on the interface used: metadata template and form based system.

The metadata template is a text file. No matter which data element within the metadata standard is required, the metadata template contains completed set of compound elements and data elements with sectional indention ready for the users. Using this template, users should have well understanding about the metadata standard in order to populate or depose appropriated metadata elements for particular geospatial data. In addition, a text editor is required to retrieve the document from the file system for editing and updating. Usually, without program control, there is a high risk of making mistakes on both data entry and document syntax. As a result, intensive syntax checking becomes a non-trivial task for users who use metadata template. Examples of this type are available from the FGDC web site and a complete template with tabbed indent is also included in the Appendix III of this thesis.

On the other hand, the form based file-store system provides relatively more functionality regarding metadata entry. This type of system comes with a user-friendly interface, which helps users to enter appropriated metadata. Usually, online help with CSDGM content is part of the components in this kind of system. The entire document syntax is controlled by program. In this case, time spent on manual data validation can be minimized or eliminated. Since it is using text file for storage, the system can only provide functionality limited to data entering and viewing. In each discrete file, there is no linkage exists between each section. Therefore, it is necessary for user to recreate data items whenever come across repeatable supporting sections such as "Citation Information" and "Contact Information". Within the same metadata document, adding repeatable data items become an operation of retyping data field by field from scratch. However, this type of metadata management system is simple and reasonable for users who just want to create and maintain single metadata document. Examples of this system type that uses plain text file as metadata storage include CorpMet95 [USCE, 1997], tkme [USGS, 1997], xtme [USGS, 1997].

2.2.2 Using RDBMS for Storage

When handling larger numbers of data sets, Relational Database Management Systems (RDBMS) based metadata management system provides more functionality and flexibility to handle data. Example systems of this type includes MetaMaker 2.0 [NBII, 1997] and Metadata Management System [LCRA, 1997]. Although the interface of those systems might look like the previous form based one, RDBMS based system provides better data management. Using this kind of system, users can recall stored metadata sections by making a selection from a list box. Inherited from the RDBMS, the system provides additional capabilities such as data indexing and querying that cannot be found from the file based system. In some of the implementations, it also provides automatic loading of metadata from an online GIS database done by add-on program script. Examples of this type like Document AML [ESRI, 1997] and FGDCMETA [INRGDC, 1997]. Since there are data elements within the metadata standard that cannot be found from GIS database, automatic operations cannot be done for the whole metadata, instead it only works on data item such as retrieval of bounding coordinates, G-polygon, attribute and so on. Other data items such as "Identification information", "Contact Information", "Citation Information" are required manual data entry.

Design and implementation of the metadata standard using RDBMS is not a easy task, since tree-like metadata structure is hierarchical in nature with interconnected content. This requires dissembling the structure into tables and joins in order to fit into the relational model. As a result, both the metadata structure and its semantics are lost. To make the system fully compliant to the standard, data or semantics might need to be duplicated in the design. Defining tables and joins for modeling tree structure is not much difficult. As shown in Figure 2.2, which gives the tables and relationships for implementing the identification section of the CSDGM using Microsoft Access. This was developed in a trial implementation of CSDGM using RDBMS in our study. The tables shown in the example are fully normalized. For each table, it provides an entry for identifying data element and maintaining the relationship between parent element and its ancestors. In fact, this is not presented in the standard and is purely for fitting it into the relational model. In addition, not all the elements in the section are mandatory but some are optional. If the optional element is not used, corresponding column of the table will be left blank. Therefore, it may find many holes in the table and this is a waste of space.

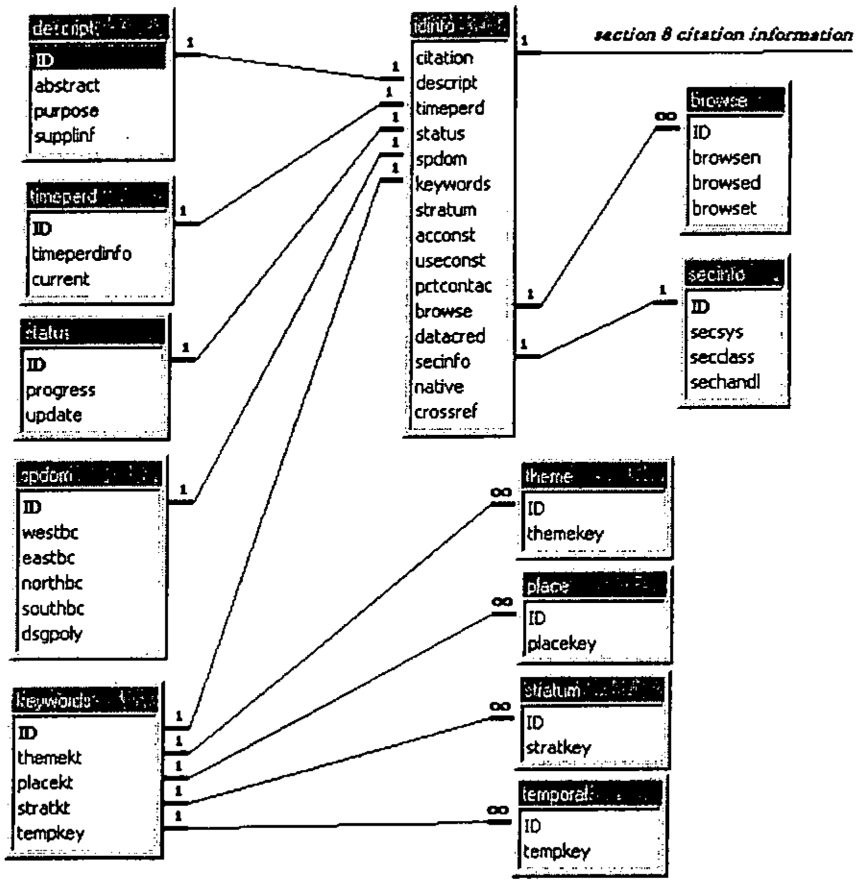


Figure 2.2 An example of tables and relationships to implement the Identification section of CSDGM

Although, it is possible to retrieve data in a hierarchical sense, it is a complex task. Practically, structure query language (SQL) can be used to build query for retrieving data in the hierarchical structure. Nevertheless, this operation will use join intensively and could degrade the system performance severely. Therefore, a better indexing system or a special querying function should be introduced to minimize the problem. Another way to relax the complexity is to de-normalize the table. Hence, this can help to decrease the number of joins in the design and solve the problem. Nowadays, several commercial RDBMS packages are now supporting hierarchical query. For instance, ORACLE, one of the well-known RDBMS in the market, is using "START WITH" and "CONNECT BY" clause to report different branches of a tree [Koch and Loney, 1997]. For each data element, a pair of unique identifiers should be assigned to it implicitly in order to maintain the child and parent relationships. These relationships are likely to be stored in separated tables and hence some data might be duplicated.

Currently, several packages using RDBMS to manage geospatial metadata are available in the community. In these packages, some tables are not fully normalized that may be because the relational modeling method is not a compatible solution for building logical model of the metadata structure.

2.2.3 Using OODBMS for Storage

As we discussed before, metadata standard document is a tree structure, starting with the top-level element i.e. metadata, and branches out to different sections. For each section, it contains compound elements and data elements. For each of these elements, it gets their own production rules, domain value and relationships among other elements. Each element, no matter it is compound element or data element, can be identified as individual objects in which its characteristics can be maintained by methods and data among the objects within the metadata document model. In this case, an object-oriented approach is well suited to model and implement metadata storage, since it allows defining new object type for describing the metadata structure. Without dissembling the original structure, and losing semantic that we face in relational modeling, object-oriented modeling approach is much more appropriate for presenting metadata structure such as storing variable-length data, and inter-connected data.

In addition, element browsing and searching on those interesting collection of elements within the structure becomes easier by traversing along section or compound element branches. The same operation that performs the same traversing in relational tables is required to use joins. This is one of the known factors that will degrade system performance. With the use of Object-Oriented Database Management System (OODBMS) to implement and build the model of metadata document. Users would obtain both the functionality found in RDBMS such as data retrieving, querying and editing, and also have better modeling and management of the data. However, present implementation of metadata management system OODBMS is not yet been found in the community.

2.2.4 Metadata Utilities

There are two metadata parsers developed by the U.S. Geological Survey (USGS), which help to validate the format of a metadata text document. A parser is a computer program, which can receive input in a form of a sequential source such as a text file. It then break it into parts according to a pre-defined model it is supposed to conform to [McGrath, 1998]. The "metadata parser" (mp) is a parser that can compare CSDGM metadata documents against the CSDGM document model. Also, there is a program called "chew and spit" (CNS), which is a metadata pre-parser. Both are built to help users to check document syntax during metadata production.

2.2.4.1 The Metadata Parser

The metadata parser (mp) is a quality control tool for metadata production and developed by Peter N. Schwetizer of USGS [USGS, 1999a]. It is a freeware and available from the USGS web site. Besides, this utility can produce the following four different formats after parsing a errorless metadata document: Standard Generalized Markup Language (SGML), HyperText Markup Language (HTML), text (TXT), and Directory Information Format (DIF). Users can then make use of these formats to publish, transfer, and share information among different departments and organizations. The following is the syntax of the command to use the mp executable:

```
Usage: mp [-e efile] [-c cfile] [-t tfile] [-h hfile] [-s sfile] [-d dfile] input_file
                        where
                        efile will contain syntax errors
                        cfile will contain configuration info
                        tfile will contain text output
                        hfile will contain html output
                        sfile will contain sgml output
                        dfile will contain DIF output
Example: mp –e GEO.err –t GEO.txt –h GEO.html –s GEO.sgml –d GEO.dif GEO.txt
```

Listing 2.1 Command syntax of the metadata parser

The above example will output five files including a syntax error file, a text file, a html file, a sgml file and a dif file for distribution purposes. However, if there are too many errors that were found in the input file, all output files will not be created except the syntax error file.

2.2.4.2 The Metadata Pre-parser

CNS is a pre-parser for formal metadata designed to assist users for converting metadata document that cannot be parsed by mp into document that can be parsed by mp. It takes in poorly formatted metadata file and, optionally, a list of element aliases, then outputs a metadata file that can be read by mp as well as a file contains lines that CNS could not figure out. This utility is also available from USGS web site and it is also free to download [USGS, 1999b].

We have been reviewed throught the management system and validating programs for the production and storage of geospatial metadata. The following sections will introduce another system for distributing and publishing of geospatial metadata document in the Internet.

2.3 Metadata Information System

A Metadata Information System is used to serve the metadata information within an interconnected network. In general, it contains two sub-components: a server and a client, which combine to forms a mechanism for delivering and accessing information.

In the following sections, we will give general idea of how does information system works in the inter-networked environment. We will then introduce the technology to build common Internet Information System, and the reason why it is not adopted for building Metadata Information System. Next, we will introduce the technology to build the metadata information system and geospatial data clearinghouse. Finally, we will focus on a new technology that we propose for distributing geospatial metadata.

2.3.1 How does the information server work?

The technology applicable for building a Metadata Information System is similar to that for building an Internet Information System. Although they both rely on the Internet protocol suites, they use different kinds of application protocol for building application specific messages. TCP/IP is a layered set of protocols for ensuring network connection and enabling cooperating computers in a network. It is designed for building the Internet and consists of two parts: Transmission Connection Protocol (TCP) and Internet Protocol (IP). TCP is responsible for ensuring reliable connection, while IP is responsible for finding route from origin to destination. In fact, all these are developed based on the TCP/IP Layering Model.

2.3.1.1 The TCP/IP Layering Model

The TCP/IP Layering Model, which is also called the Internet Layering Model or the Internet Reference Model, describes network communication between machines in term of five layers [Douglas, 1997]. This model is developed based on the Open Systems Interconnection (OSI) 7-layer Model, therefore, some of the layers defined in the TCP/IP Layering Model could be found from the OSI model. In the TCP/IP Layering Model, each layer is dedicated to a particular purpose, related to either software or hardware. More precisely, one could define the communication protocol with a series of services, one for each layer. To send and receive message from one computer to another computer, a message will go through each layer. For each layer, which receives the message, it would call services available from either its successive lower layer or upper layer that depends on the action whether it is sending or receiving the message. Nevertheless, the end point of the message will reach the top layer for network software to further interpret the message contents. In fact, this model helps to simplify the complex network communication into smaller and easier pieces, which in term make it easier for development. Figure 2.3 shows the five layers of the TCP/IP reference Model.

As shown in Figure 2.3, to enable software accessing physical network or vice versa, there are five layers: Application, Transport, Internet, Network interface and Physical. The top layer is called application layer, which defines messages corresponding to particular application running in the Internet. Below is transport layer, which specifies how to ensure reliable transfer and this is done by the TCP. The third layer is called Internet layer, which is done by the IP. The lowest layers, network interface and physical layers are closely related to hardware, which organize message into special format and transmit it into the physical network.

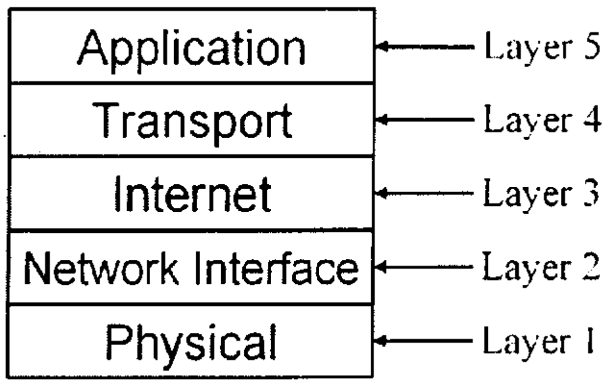


Figure 2.3 The five layers of the TCP/IP reference model. [Douglas, 1997]

According to this model, network software will generate a message according to the particular application layer protocol. The message will then be passed to lower layers and finally to the physical network. For each layer, it will add information in the header of the message for special purpose. When another machine receives the message, it would pass the message from the lowest layer to successive higher layer with the same reference model. For each layer, it will remove additional information until the network software receives the original message. This is the whole mechanism for passing information across network.

2.3.1.2 The Application Protocol

There are different kinds of application protocols available and they are designed for different applications. However, in the TCP/IP layering model, application protocol runs on top of TCP/IP, in which the application layer defines the message and the TCP/IP carries and sends the message to the destination. For example, there is File Transfer Protocol (FTP) for file transfer, Simple Mail Transfer Protocol (SMTP) for mail services and telnet for remote login. FTP, SMTP and telnet are application protocols for different applications. For information distribution and sharing, there is Hypertext Transfer Protocol (HTTP), which is the most popular application protocol working in the Internet. For providing cataloging searching and retrieval, which are applied to build geospatial data clearinghouse and library catalogue system, there is Z39.50 Information Retrieval Protocol.

Distributing metadata document in the inter-networked environment, the metadata information system also requires a protocol for communication. Since it is running in the Internet, it uses TCP/IP to ensure reliable network connection and requires an application protocol for defining the message content. In fact, there are two basic criteria for selecting the protocol such as information browsing and precise searching. In the following sections, we will explore the use of two application protocols that are designed for information distribution.

2.3.2 Hypertext Transfer Protocol (HTTP)

According to the specification of HTTP [Fielding et al., 1999], HTTP is a request/response and stateless protocol. A client sends a request to the server in form of a request method, a uniform resource identifier, and protocol version, followed by a message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including protocol version of the message and a success/error code, followed by a message containing server information, entity metainformation, and possible entity-body content. For each connection, it is a pair of request and response action, in which each pair of action is independent of all others. There is no information for tracking user activities and hence there is no connection status available. Nevertheless, there is a mechanism called "cookie", which is used to remedy this limitation. A cookie is a special text file that HTTP server puts on a client's local disk. By reading the file, the server could keep track of user activities. However, it is a user decision to allow the cookie to be saved to enable the server to provide better services.

Most HTTP communications are initiated by client software in which each communication consists of a request to be applied to resource on some origin servers. In the simplest case, this may be a single connection between the client software and the origin server. A more complicated situation occurs when one or more intermediaries are present in the request/response chain. There are three common forms of intermediary: proxy, gateway, and tunnel. A proxy is a forwarding agent. A gateway is a receiving agent and acts as a layer above some other server(s). If necessary, it will translate the requests to the underlying server's protocol e.g. HTTP to Z39.50 gateway. A tunnel is a logical path through which encapsulated messages travel through the Internet. Once the messages reach their destination, it would be unencapsulated. This method should be used with tunneling protocol, which provide routing information for creating the encapsulated message. In addition, it is usually used when secure connection is required between two points in the Internet.

HTTP communication usually takes place over TCP/IP connections with the default port set to TCP 80, though other ports can also be used. The port number is used to keep track of the activities of particular application. In fact, to distinguish different users using the same port, there is a random number assigned before the port number for each connection. Implementation of HTTP is an Internet information server or a web server, which usually uses with Hypertext Markup Language (HTML). Examples of Internet information server are Microsoft Internet Information Server, Apache Web Server and Netscape Communication Server.

2.3.2.1 Hypertext Markup Language (HTML)

Working with HTTP, Hypertext Markup Language (HTML) is a content-based language [Raggett, 1997]. It enables information publisher to define how information would be presented in their client software such as Netscape or Internet Explorer. In addition, HTML is an application of Standard Generalized Markup Language (SGML), which is a standard system to define document structure. From the specification of HTML, it defines a number of markup tags dedicated for document presentation. For instance, if users want to display a word "Metadata" as document header in bold and larger size for emphasis purpose. One can use a standard markup tag pairs: start tag "<h1>" and end tag "</h1>" enclosing the whole word i.e. "<h1> metadata </h1>". However, only some of the markup tags require end tag. The client software that understands HTML will adjust the text format accordingly such as bold format, and the tags would be hidden from normal view i.e. **metadata**. There are other markup tags for different formatting purpose such as <li> means list item, <p> means paragraph and so on.

In addition, there is a special type of markup tag provides hyperlink. A hyperlink defines a link between documents within the Internet. By clicking the hyperlink with mouse pointer, current document will be changed to another document referred. This important feature enables users to traverse and browse among documents with similar topic in the Internet. However, it is not suitable to use for searching, since only related documents will be linked together. To perform searching, some web search engines such as AltaVista, Yahoo, InfoSeek can be used. In fact, there are databases that store and make indices for a huge amount of html documents found in the web. It allows users to type a keyword and query into the engines. If the search is successfully completed, a list of hyperlink will be displayed for further selection.

2.3.2.2 The Use of HTML and HTTP for Metadata Delivery

Thinking about application for serving geospatial metadata information, the use of HTML would limit the information discovery process. As we mentioned, HTML only supports a limited set of markup tags and user-defined markup is not possible. This lead to several severe problems in regards to its extensibility and use for information searching capability. A document written in HTML has no capability to deliver content meaning to users, since HTML is primarily designed for machines to interpret the document appearance and link related documents together for faster information browsing. Since there is too much information stored in form of HTML, it is easy to find similar documents but not the precise subject of the search. Besides, information content in this case is difficult or impossible for machine to understand except it has intelligence to interpret and determine the contents meaning within each presentation markup. When users try to search information for particular thing using the web search engines, any hyperlink found that contains the input keyword will be drawn, no matter it is related to the topic or not. For example, if one wants to make a search using keyword "Hong Kong Map" from the Internet, the result is properly a list of hyperlink that are HTML documents contain all or part of those words. It is not easy to obtain accurate result in this sense. This problem becomes worse when the users use a number as searching keyword, since a number means many things such as size, length, latitude, longitude, if one does not specified.

Regarding the use of HTTP, it is a request and response protocol. Although it is a major protocol working in the Internet, with HTML, it is not sufficient to be used for delivering geospatial metadata because it lacks database-like capabilities such as search and retrieve.

2.3.3 ANSI/ISO Z39.50 Information Retrieval: Application Service and Protocol

At the time FGDC was seeking a solution to overcome those mentioned problems, it was found that ANSI/ISO Z39.50 Information Retrieval Protocol (ANSI Z39.50/ISO 10163) fulfilled the requirements for distributing context information like the geospatial metadata [Nebert and Fulton, 1997]. Z39.50 is composed of application service definition and the protocol specification, which are dedicated for information retrieval. It is an application protocol, which corresponds to the TCP/IP Layering Model and originates from a library standard for cataloguing book records for many years. It was built to satisfy a requirement for searching and retrieving USMARC-formatted bibliographies records [LeVan, 1990].

However, Z39.50 is not a replacement of HTTP or vice versa. Since it is a fact that HTTP usage dominates the Internet usage and the combination of HTTP and HTML is recognized to give a good and easy information browsing capability for users. Besides, both protocols are designed for different application. Instead, Z39.50 is a complement of HTTP via an HTTP-Z39.50 gateway. Z39.50 is a "stateful" protocol. Stateful means that for each connection to a Z39.50 server, there is a persistent session between starting and closing a connection. This session oriented connection allows clients iterative refinement of search result sets and multiple record retrieval requests against the same result set [LeVan, 1990]. In addition, this also enables the server to monitor users activity. Stateful protocol also provides a higher efficiency in communication over the network, since overhead on opening and closing a session is minimized for each connection.

2.3.3.1 The Z39.50 operation

According to the Z39.50 specification [ANSI/NISO, 1995], to make a connection from an origin, i.e. client, to a Z39.50 server, requires establishing an application association or A-association using TCP/IP. With the A-association established, the client sends an initializing request to the server in form of an initial method, a series of optional switches, user id, password and so on. The server responds with a status line including a series of optional switches replied and a success/error code. If the connection is successful, a Z-association or Z39.50-assoication is established and the client sends a search request to the server including a record set id, database name, and a query. Meanwhile, the client can make other Z-associations within the A-association. Again if the search operation succeeds, the client sends a present request to draw results from the server. A present request is a mechanism initiated by the origin to retrieve search results from the server. Before the Z-association is closed implicitly by either client or server, multiple search and present request can be made. If the A-association is closed, all the Z-associations will also be closed. In addition, within a Z-association, if there is any operation problem found, the server will send diagnostic information to client. Then, both the server and client could continue to negotiate by sending multiple messages and help to find resolution to solve the problem.

Similar to the HTTP, Z39.50 communication is usually initiated by client software. The initialization could comprise a series of requests to be applied to resources on multiple servers. For each message that is ready to transmit from either side, it should be formulated according to Abstract Syntax Notation 1 (ASN.1) grammar and be encoded by Basic Encoding Rules (BER). The ASN.1 is an ISO standard i.e. ISO 8824, which is used for defining message content [LeVan, 1990]. While BER is another ISO standard i.e. ISO 8825, which is used for defining records as composed of a triple of values: a tag, a length and a value i.e. TLV. [LeVan, 1990]

In the simplest case, a Z-connection may be accomplished via a single connection between the client software and the server. A more complicated situation occurs when a client uses HTTP to access the server. A Z39.50 to HTTP gateway can help to translate the requests between Z39.50 protocol and HTTP protocol or vice versa for communication. Z39.50 communication usually runs over TCP/IP connections, which is similar to the HTTP. The default port is TCP 210, but other ports can be used.

2.3.3.2 Defining GEO Profile in Z39.50

The implementation of Z39.5 permits user-defined profile for particular application. A profile is an analogous of a template, which provides any necessary attributes; tags as well as associated operators can be freely defined for a particular community or an application. Besides, all the newly developed profiles could start with the bib-1 profile for ready to use attributes and definition. Bib-1 profile is originally defined for library cataloguing uses. Based on CSDGM and ANSI/NISO Z39.50, the FGDC developed geospatial metadata profile known as "GEO". This provides Z39.50 specifications for geospatial metadata application and states an Attribute Set for searching, an Element Set Names for server presentation of results, and prescribes the formal Record Syntaxes to be supported by GEO servers for the transfer of records [Nebert, 1999]. To incorporate CSDGM into the GEO, each metadata element is associated with three attribute types: Use Attribute, Structure and Relation. The Use attribute is an integer assigned for identifying each metadata element, while there is another tag name assigns to the integer for interpretation. The Structure attribute is data type for each metadata element, while the Relation attribute is a series of operations for searching purpose uses with particular Structure attribute. For example, one of the data elements in CSDGM, "East Bounding Coordinates", is encoded as a GEO attribute type "eastbc" within the GEO attribute set. For each GEO attribute type, it consists of attribute value and structure type identified by two integers such as "2039 109" in case of "eastbc". The first integer tells the element's attribute type is "eastbc" and the second integer tells the structure of that type is "Numeric String", while numeric string is character string that represents a number. For each structure type, there is a series of operators called "relation attributes" specified in the Z39.5 profile and specification used for information searching purpose. For instance, structure type "Numeric String" contains operations (relation attributes) such as Less Than, Less Than or Equal, Equal, Greater Than or Equal and Greater Than. Table 2.1 shows the GEO attribute set for the East Bounding Coordinate of the Identification section in the CSDGM.

East Bounding Coordinate	
Use Attribute	eastb (2039)
Structure	Numeric String (109)
Relation	Less Than (1), Less Than or Equal (2), Equal (3), Greater Than or Equal (4), Greater Than (5)

Table 2.1 Summary of the GEO Attribute Set for the East Bounding Coordinate of Identification Section in the CSDGM

Following the specification of GEO profile, Z39.50 information retrieval services and protocol, a metadata information system or Z39.50-GEO system can be built. This system consists of client and server program providing capability in geospatial information searching and retrieval. One of the Z39.50 – GEO implementation examples is Isite [Gamiel, 1994] from the Clearinghouse for Networked Information Discovery and Retrieval (CNIDR).

2.3.3.3 Isite – The Metadata Information System

Isite is basically an Internet information system that employs Z39.50 as its application protocol. It can be used as a Metadata Information System if it is configured with the GEO profile. However, it could be used with any other profiles. It is developed and maintained by the CNIDR. Isite provides utilities to support text file indexing and searching in which users are required to re-index the entire database once metadata document was updated. There are several components in Isite that include the Z39.50 communications applications, HTTP to Z39.50 gateway, a text indexing system called Iindex, and a text search system called Isearch. To enable developers with extensibility, a Search API (SAPI) is provided and it can be used to develop connection between database systems that Isite does not support. The whole package of Isite is a freeware and can be downloaded from the CNIDR web site.

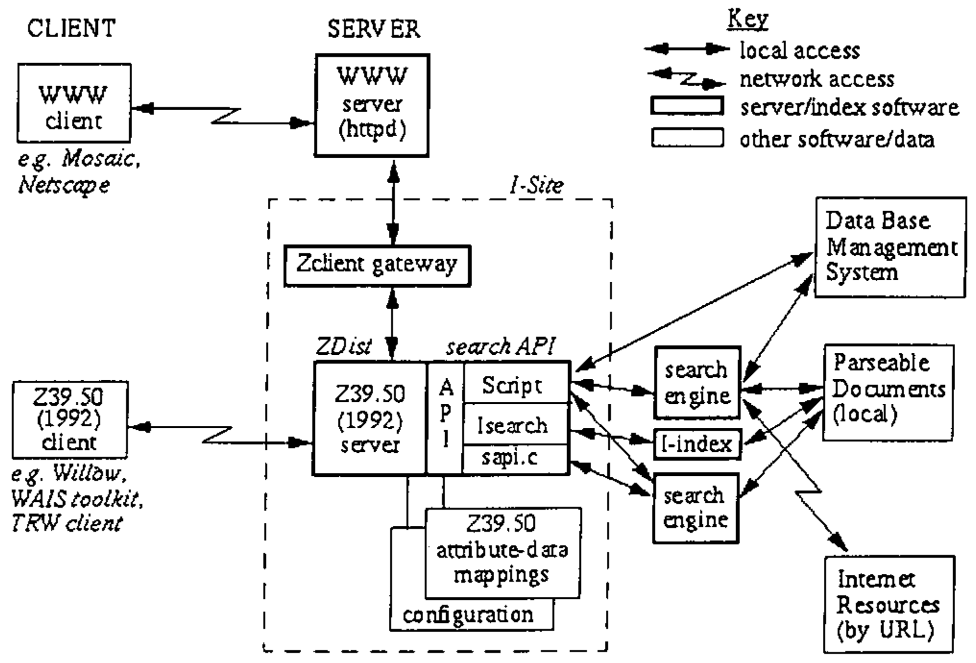


Figure 2.4 Architecture of I-Site Z39.50 software from the Clearinghouse of Networked Information Discovery and Retrieval [CNIDR, 1997]

The core of Isite is a Z39.50-1992 service that was designed for accepting a request from a Z39.50 client and translating the search request through the search API to one or more local or remote stores of information, and returning a list of relevant documents. In Figure 2.4, it shows the architecture of Isite and following is a brief description of each component in Isite.

2.3.3.3.1 Storage system

Isite uses an ordinary file system to store metadata documents. Any text document such as HTML, SGML, TXT and DIF can be put into Isite. However, if users want to use advanced database system such as ORACLE, there is a search API (SAPI) that provides the capability to develop an interface between Isite and the database system, i.e. RDBMS, OODBMS. However, Isite uses flat system for storage and database system is not readily available in the package.

2.3.3.3.2 Indexing

Indexes improve the efficiency of searching metadata. As part of Isite, there is an indexing program called "Iindex", which can index text according to the specified document type (doctype) used. A doctype is a module that explains how to find logical documents and sub fields or tags within those documents. For instance, the "SGMLTAG" doctype tells Isearch how to find the "<title>" tag inside an SGML document.

2.3.3.3.3 Searching

Isearch is a searching utility within Isite. It uses the results generated by Iindex to make efficient searches on documents. Isearch supports two types of Boolean query: Infix notation Boolean query and Reverse Polish Notation (RPN) Boolean query. The RPN Boolean query is one of the required features of Z39.50 compliant server.

2.3.3.3.4 Search API

Search API (SAPI) that comes with Isite supports free-text indexing and searching of text documents with a command-line based search protocol (Script). The search protocol allows one to define a search script to pass along query terms and perform retrieval from a database or other organized collection of information. This is a tool originally designed for testing the Search API. A simple C-based API for direct software integration is available for these basic functions (sapi.c) to enable programmers to make direct connections into databases.

2.3.3.4 The Z39.50-GEO Client

To enable browsing and searching for metadata, a Z39.50 – GEO compliant client is required. This is part of the metadata information system dedicated for Z39.50-GEO standard. Using the client software, users can submit query with specified attribute and attribute value to a Z39.50 – GEO server. Presently, there are several approaches for a data provider to enable users having Z39.50 - GEO searching capability in the Internet. One of the approaches is to develop a JAVA applet for the web browser. JAVA applet is a special kind of JAVA program that can be downloaded and run automatically in a web browser. A JAVA-enabled web browser can use its own JAVA interpreter to interpret bytecodes or class files and run the program [Cornell and Horstmann, 1998]. One of the implementation examples using JAVA applet can be found in the FGDC web site.

Another approach is to use Z39.50 to HTTP gateway. A gateway is a program providing a layer between two different protocols. In general, it is transparent to client, and it helps to translate the different protocols, in our case from HTTP to Z39.50 or vice versa. With the gateway, users can use an ordinary web browser to do searching and information retrieval.

2.4 Geospatial Data Clearinghouse

When users are looking for geospatial data, it is desirable to have a single contact point for accessing all the available resources. Data clearinghouse is a concept to provide this capability by connecting all the available Z39.50-GEO servers together to form an information network.

Conceptually, a clearinghouse is a sort of agent, which serves as an intermediate point for exchanging things such as information, between parties. The role of clearinghouse is to ensure quality, integrity and operation of any exchange as well as the expected items delivered. In the same sense, geospatial data clearinghouse is an data exchange focal point, which aims to guarantee data quality, as well as increase the usability of data by providing easier data accessing interface. Besides, it also provides a function to share data and helps to cease duplication of geographic database creation works. In the view of implementation, data clearinghouse can be interpreted as a software interface for serving geospatial data exchange. Behind this interface, it may be a huge resource of geospatial data.

In 1994, the US government initialized the development of National Spatial Data Infrastructure (NSDI) coordinated by FGDC. "The NSDI encompasses policies, standards, and procedures for organizations to cooperatively produce and share geographic data. There are 15 federal agencies that make up the FGDC develop the NSDI in cooperation with organizations from state, local and tribal governments, the academic community, and the private sector." [FGDC, 1997a]

To develop the NSDI, FGDC has developed a metadata standard detailed in "Content Standard for Digital Geospatial Metadata, CSDGM" as mentioned in previous sections. This helps to standardize the data storing method in distributed clearinghouse node [FGDC, 1997b], and hence, the data clearinghouse helps to enforce the goal of geographic data sharing. This is also a foundation for developing and managing geospatial metadata.

According to FGDC, the existing data clearinghouse is a decentralized or distributed system of servers located in the Internet. Each server contains field-level descriptions of available digital geospatial data. This descriptive information is geospatial metadata. It is collected in a standard format to facilitate query and consistent presentation across multiple participating sites. Clearinghouse uses readily available Web technology for the client side and uses the ANSI standard Z39.50 for the query, search, and presentation of search results to the Web client.

As described by Nebert from FGDC [Peng and Nebert, 1997], a fundamental goal of data clearinghouse is to provide access to digital geospatial data through geospatial metadata. The clearinghouse functions as a detailed catalog service with support for links to geospatial data and browse graphics. Clearinghouse sites are encouraged to provide hypertext linkages within their metadata entries that enable users to directly download the digital data set in one or more formats. Where digital data is too large to be made available through the Internet or the data products are made available for sale, linkage to an order form can be provided instead of a data set. Through this model, clearinghouse metadata provides low-cost advertising for providers of geospatial data, both non-commercial and commercial, to potential customers via the Internet. This implies that data clearinghouse can be used as a commercial tool for data product advertising.

As further description of data clearinghouse by FGDC [FGDC, 1997a], it allows individual communities to band together and promote their available digital geospatial data. Servers may be installed at local, regional, or central offices, dictated by the organizational and logistical efficiencies of each organization. All clearinghouse servers are working in a peer network model within the clearinghouse activity without hierarchy among the servers and permitting direct query by any user in the Internet with minimum transactional processing. In other words, users can access all the resources by connecting to any server without worrying about its contents.

2.4.1 Centralized System or Distributed System

As mentioned in the previous section, the NSDI builds clearinghouse in a distributed manner with plenty of supporting resources. However, in some situations due to limitation of personnel and hardware resource, building of data clearinghouse network using the distributed approach may not be a practical solution. Centralized approach for building a data clearinghouse may be an alternative solution. Therefore, to which approach is selected for implementing data clearinghouse network, it depends on the available resources and other conditions such as the frequency of data update.

To build a centralized data clearinghouse network, a database will store all the geospatial metadata in a single location. In this case, the control of metadata quality as well as data delivery can be ensured, the operational personnel can be minimized. Therefore, the operational cost can then be reduced. However, data updating procedure might take longer time to complete. Metadata in this case stored in the server might not be synchronized to those data in the GIS database running at different sites, unless a controlled maintenance procedure can be introduced to enforce and surely maintain the currentness of stored data.

In distributed database approach, metadata will store in multiple servers that are distributed located in a territory. For each server, it might solely store geospatial metadata or it might share resource from the same GIS database as well. In the latter scenario, metadata can be replicated with the online geospatial data automatically, which maintains an expected currentness of information. Within a distributed data clearinghouse network, remote client can access and browse data from one of the servers. Once, the query is passed to the server, the server will then submit the same query to the other servers for feedback, which is technologically feasible to do so. In addition, the selection of which servers to be accessed is user definable. Different from the centralized approach that could use proprietary communication protocol, the distributed one should strictly conform to a communication standard, i.e. Z39.50.

Nevertheless, for small organization, the distributed approach is not an economical way to be implemented, since it might cause overloading of the technical support and operation in a local site. Although this may greatly increase the data currentness by eliminating the data transfer time, from the viewpoint of cost benefit, centralized approach in this case is a better solution.

For both approaches, clients are distributed and have the capability to access the server remotely. It should be built as lightweight as possible, so that most of the processing works can be done on server side. In fact, both approaches might work for building a data clearinghouse. The selection of which approach to follow depends on the cost benefit and the level of data supply, i.e. currentness of the metadata, that the organization wants to provide, while combination of both approaches might work in different levels of organization. For instance, at province or state level, it may use distributed approach; at local government level, it may use centralized approach.

2.4.1.1 The Hong Kong Situation

Regarding the Hong Kong situation, the Government adopted to use CSDGM as metadata standard for documenting GIS database in the region. Since the territory is in city level, even data updating is very frequent the time for transferring data from one place to another would not take a long time. With the completion of various GIS and mapping systems in recent years, geospatial metadata is starting to be recognized that enables geospatial data users to think about how to exploit geospatial data sets. In addition, even there is various systems are running, geospatial metadata can be maintained at a very high level of currentness, because most of the them are base map system. The metadata contents are static and would not be changed by daily data updating works. For new data set, it is expected to be the data producers' responsibility to prepare it.

In addition, there is a lack of centralized mechanism for collecting, managing, and sharing local metadata documents to public. Inevitably, this mechanism helps to enable data producers or providers to increase the usability of their data sets, promote the usefulness of the geospatial data, while allows public to browse it in a cost effective and controllable manner. Although, the awareness of the importance of geospatial metadata in Hong Kong is increasing, the mechanism for sharing geospatial information is still under development. In fact, without the concrete setup for the metadata production, further application on exploiting geospatial metadata is hard to develop. Nevertheless, metadata management is a new topic and trend to geospatial data management; it is foreseeable that this will provide a significant function in the GIS community.

2.4.2 How does the National Geospatial Data Clearinghouse work?

The National Geospatial Data Clearinghouse (NGDC) developed by FGDC uses distributed database approaches to manage metadata. In order to make a standard searching attributes among multiple servers and providing field-searching capabilities, FGDC uses Z39.5 search and retrieval protocol as an information-searching media.

According to FGDC, the Z39.50 protocol includes client and server software that establish a connection, pass a formatted query, return query results, and present identified documents to the client in one of several formats. The formats include HTML, SGML, text as well as DIF. On the server computer, Z39.50 server software typically communicates with a search engine (database or indexing software) to process the query and formulate the results.

According to Z39.50 specification, the FGDC has developed a profile for geospatial metadata, called "GEO," which provides guidance on how to implement FGDC metadata elements within a Z39.50 service. In the GEO profile, it reuses part of Elements from the base bib-1 attribute set which are also used in the U.S. Government Information Locator Service (GILS) profile.

Indeed, the Z39.50 – GEO provides a set of known sets of parameters that are represented as numeric-tag attributes. Those sets of parameters are actually developed from the metadata standard by assigning an attribute type and numeric-tag to each metadata element. For example, when the client system wants to make a query of "Title", it will translate the query object "Title" to a know identifier, that is called "4" in this case and pass it to a Z39.50 server for information searching. Hence, the server side recognizes numeric-tag, and does know how to make query using either an index file or a database.

In previous sections, we have introduced the basic idea to build and operate metadata information system. We have also discussed the pros and cons of using HTTP and Z39.50 in the Internet. In addition, we have introduced the concept to build the geospatial data clearinghouse. In the following section, we will focus on a new Internet technology called Extensible Markup Language (XML). We apply this technology to build application model for helping users to find appropriate geospatial processing knowledge and data resources.

2.5 Extensible markup language (XML)

In this section, we will introduce a new standard for defining next generation markup language. It is called the extensible markup language (XML). In general, XML is a standard for defining markup language and it is a subset of Standard Generalized Markup Language (SGML) [Bray et al., 1998]. In the coming sections, we will introduce the reasons why the web community developed XML and describe the mechanism by which it can benefit the present and future demands of the community. Next, we will explore the structure of XML and the mechanism for using XML. Then, we will describe the document object model (DOM), which is a standard for building software tool for manipulating document such as XML document as objects [Apparao et al., 1998]. Finally, we will review possible applications of XML that would help to provide simple and efficient sharing, distribution and understanding of geospatial-related information among the geospatial community.

2.5.1 Initiatives of XML development

Prior to XML, SGML is the only standard to defining document format. It has a lot of powerful features to define and format document [Bray et al., 1998]. However, SGML is too complex and sophisticated to be implemented for simple application such as web application [Clark, 1997]. This is the main reason why

SGML is little known to the public and rarely used to build low cost applications. To open the power of SGML to public, it should be simplified, so that it is easier to implement, understand and more suitable for delivering data over the Internet.

On the other hand, Hypertext Markup Language (HTML), which is an application of SGML and currently the most widely adopted document type in the Internet. It has several advantages to make it popular. For example, its structure is simple, it is easy to implement and supported by much freeware and commercial software tools. However, it has serious limitation to describe information contents. This limitation consequently blocks the opportunity to manipulate and generate useful information from the document itself. Besides, the following are shortcomings of HTML:

- HTML is not extensible.

  As defined by SGML, HTML has a limited set of markup tags available for users to create their own HTML document. It is not feasible for users to define their own tags, since there is no browser capable of understanding the user-defined tag and to render it properly. On the contrary, software vendors have the capability to modify or introduce new tags and this would result in generating non-standard tags that makes HTML programming become difficult. For example, Microsoft introduces HTML tags, supported by Internet Explorer, but Netscape Navigator cannot understand and render it properly. This problem also affects the feasibility to use HTML for encoding geospatial metadata. If HTML is extensible, this is not a problem. In fact, there is a big demand from both the vendors and users to extend markup tags for better document description.

- HTML is designed for data rendering only.

Most of the HTML tags are designed for data presentation and formatting use. Other tags are designed for hyper-linking. HTML browser understands how to present HTML document according to different tags used. However, it does not understand the data contents inside. In addition, HTML document can only provide a single view of data. It is because information about presentation style and the data contents is merged into one piece, which is indivisible. The way to render data is fixed and this cannot be changed before or after browsing. What is expecting from the community is that data or presentation style could be changed according to different circumstance when the information is being browsed. For example, GIS users who check for available geospatial data set by browsing a web page. Some users would like to view the sample graphic only, but other users may like to view some statistical analysis about the data set instead. In fact, they are viewing the same information with different rendering results only.

- HTML has weak internal structure.

HTML provides many tags for presenting data. In some cases, there is no strong control for keeping the logical relationship between different tags. For example, H1 as tag of book title, H2 as tag of chapter and H3 as tag of paragraph. It is possible to put H1 element inside the H2 element or H2 element inside the H3 element that cannot maintain the structure between book title and chapter, chapter and paragraph. Logical relationships in this case cannot be maintained.

- HTML does not provide semantic information about the document contents.

Without semantic information provided in HTML, it is impossible for computer program to understand the document content. Therefore, useful information cannot be extracted and this disallows data processing and manipulation in the document. One of the common examples that this problem introduces is the difficulty of Internet searching. For example, if users try to locate HTML links with a keyword "GIS" using Internet search engine such as Yahoo, AltaVista, it would return a large number of unrelated HTML document links. Most of these links are not related to what the users wanted. As a result of this shortcoming, there are numerous sites of useful links maintained by organizations and individuals attempting to help others browsing information on particular subjects. However, this is not an ultimate solution to the problem.

To solve these problems in HTML and make the SGML manageable on simple application, in 1996, the World Wide Web Consortium (W3C) formed working group and started to develop Extensible Markup Language (XML). The fundamental XML standard was completed two years later in 1998. However, development based on XML is still on going with different working groups organized on different aspects as follows [W3C, 1998].

- XML Coordination WG for providing coordination between different parties involves in developing XML and its related technologies.

    - XML schema WG for addressing the means to define structure, content and semantics of XML documents so that it helps component parts of an XML application fit together.
    - XML linking WG for designing advanced, suitable, and maintainable hyper-linking and addressing functionality.
    - XML information set WG for looking at a more effective description for XML documents in term of document tree structures, elements and attributes. The goal is to develop a common reference for describing XML and other specification could build on top of it to ensure interoperability among various XML based data and software.
- XML Fragment WG for defining a method to retrieve and manipulate fragments of XML document from the server side. Therefore, for each transaction of XML based information, client could retrieve the useful part of data without the entire document. For the retrieved fragments of XML document, it should find methods to handle it for different cases such as immediate or later viewing use, editing, accumulation or assembly of the fragments to form new information or other processing and,
- XML syntax WG for different aspects such as
    - XML style sheet linking, which define the presentation style of XML document, since XML document does not deliver formatting information,
    - XML profile, which specify particular application may require to support a set of XML features that similar to Z39.50 profile,

- XML standard control, tracking internationalization developments and errata to XML1.0.

The introduction of XML in fact helps to complement HTML but also making SGML available on simple application. The following sections will introduce the primary goal of XML.

2.5.2 The goal of XML

According to the W3C, HTML is an application of SGML for creating document in the Internet, while XML is a simplification of SGML for the Internet and it is not a replacement of HTML. The main objective of XML is enabling the use of SGML on web application. Mechanically, XML could define data or document, which is self-describing, both readable and understandable by machine and human. In addition, it has major characteristics like easy to create, easy to be processed and so on. Besides, the specification of XML also defines the working behavior of a software module called XML processor, which is used to read XML document and provide access to their content and structure. It is expected that XML processor is working on behalf of an application to process data. The following is extract from the activity statement that W3C briefly states what XML will do [W3C, 1997c]:

- Enable internationalized media-independent electronic publishing,
- Allow industries to define platform-independent protocols for the exchange of data, especially the data of electronic commerce,

- Deliver information to users agents in a form that allows automatic processing after receipt,
- Make it easy for people to process data using inexpensive software,
- Allow people to display information the way they want it and,
- Provide metadata, which is data about information that help users to find information, help information producers and consumers to find each other.

According to the description, XML is independent of language, platform and layout. In fact, XML specification is defined based on the ISO-10646 (Unicode) character set, so it is capable to handle any character available in the world [Maruyama et al., 1997]. As defined, any XML processor should conform to character encoding UTF-8 and UTF-16. However, some XML processors may include other character encoding such as European, Chinese encoding, Japanese and so on.

For the platform, it highly depends on the programming language that implements it. Currently, most of the XML development such as application or processor is developed using JAVA, which is a platform independent object-oriented programming language.

In addition, XML itself provides description solely on data and relationships between each atomic data element. Data presentation is not defined in XML, but it is defined in XML Stylesheet Language (XSL) that is a language for writing style sheet, which specifies the presentation of XML document. A single XML document could have more than one style to render the data, which depends on different circumstances. As a result, if information is expressed in XML, one can find the information much effective and precise, multi-language and multi-style. We can see that W3C would like to make use of XML to enforce the capability of data sharing, data distribution and data exchange among documents within the industry.

2.5.3 An example of XML

In order to show the significance of XML that how it can deliver meaningful and understandable contents to both users and machines, we use a small part of a geospatial metadata document. The extraction of the document will be shown in term of a HTML document with its appearance on a HTML browser view, next it will be rewritten using XML with self-defined tags. The following Listing 2.2 is an extract of a geospatial metadata document written in HTML.

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title></title>
</head>
<body>
<dl>
  <dt><em>Keywords:</em> </dt>
  <dd><dl>
      <dt><em>Theme:</em> </dt>
      <dd><dl>
          <dt><em>Theme_Keyword_Thesaurus:</em> None </dt>
          <dt><em>Theme_Keyword:</em> Superficial
             Geology </dt>
        </dl>
      </dd>
      <dt><em>Place:</em> </dt>
      <dd><dl>
          <dt><em>Place_Keyword_Thesaurus:</em> None </dt>
          <dt><em>Place_Keyword:</em> SAN TIN </dt>
          <dt><em>Place_Keyword:</em> SHEUNG SHUI </dt>
          <dt><em>Place_Keyword:</em> KAT O CHAU (CROOKED
             ISLAND) </dt>
          <dt><em>Place_Keyword:</em> CASTLE PEAK </dt>
          <dt><em>Place_Keyword:</em> YUEN LONG </dt>
          <dt><em>Place_Keyword:</em> SHA TIN </dt>
          <dt><em>Place_Keyword:</em> SAI KUNG
            PENINSULA </dt>
          <dt><em>Place_Keyword:</em> TUNG CHUNG </dt>
          <dt><em>Place_Keyword:</em> SILVER MINE BAY </dt>
          <dt><em>Place_Keyword:</em> HONG KONG ISLAND
            AND KOWLOON </dt>
          <dt><em>Place_Keyword:</em> CLEAR WATER BAY </dt>
          <dt><em>Place_Keyword:</em> SHEK PIK </dt>
          <dt><em>Place_Keyword:</em> CHEUNG CHAU </dt>
          <dt><em>Place_Keyword:</em> HONG KONG SOUTH
            AND LAMMA ISLAND </dt>
          <dt><em>Place_Keyword:</em> WAGLAN ISLAND </dt>
        </dl>
      </dd>
      <dt><em>Stratum:</em> </dt>
      <dd><dl>
          <dt><em>Stratum_Keyword_Thesaurus:</em> None </dt>
          <dt><em>Stratum_Keyword:</em> Superficial </dt>
        </dl>
      </dd>
    </dl>
  </dd>
</dl>
</body>
</html>
```

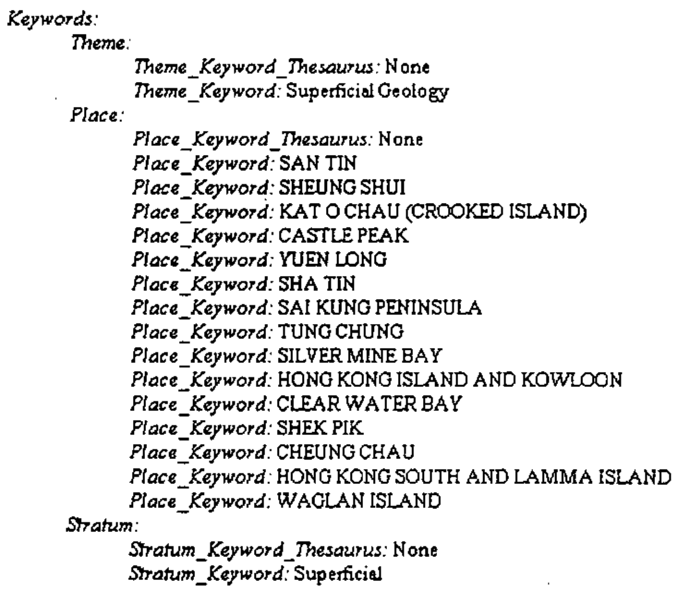
Listing 2.2 Extraction of CSDGM: The keyword element of the identification information section

This extract shows the keyword element of the identification information section. There are different tags to format the information such as DL, DT, DD and EM. According to HTML specification [HTTP WG, 1998], the first three tags are used to draw definition lists that take the form as Listing 2.3.

```
<DL>
  <DT> term name
  <DD> term definition
  ...
</DL>
```

Listing 2.3 HTML tags for definition lists

Accordingly, DT elements can only act as containers for text level elements, while DD elements can hold block elements such as paragraph as well, excluding headings and address elements. In addition, EM provides emphasis that renders the enclosed text in an italic font.



```
Keywords:
    Theme:
        Theme_Keyword_Thesaurus: None
        Theme_Keyword: Superficial Geology
    Place:
        Place_Keyword_Thesaurus: None
        Place_Keyword: SAN TIN
        Place_Keyword: SHEUNG SHUI
        Place_Keyword: KAT O CHAU (CROOKED ISLAND)
        Place_Keyword: CASTLE PEAK
        Place_Keyword: YUEN LONG
        Place_Keyword: SHA TIN
        Place_Keyword: SAI KUNG PENINSULA
        Place_Keyword: TUNG CHUNG
        Place_Keyword: SILVER MINE BAY
        Place_Keyword: HONG KONG ISLAND AND KOWLOON
        Place_Keyword: CLEAR WATER BAY
        Place_Keyword: SHEK PIK
        Place_Keyword: CHEUNG CHAU
        Place_Keyword: HONG KONG SOUTH AND LAMMA ISLAND
        Place_Keyword: WAGLAN ISLAND
    Stratum:
        Stratum_Keyword_Thesaurus: None
        Stratum_Keyword: Superficial
```

Figure 2.5 An example of HTML view by browser.

Looking at the HTML Listing of the keyword element, even without a HTML browser to render the data, it is quite clear that anyone can recognize and understand the document content. However, this is not the case for computer program, since it is impossible to identify the data value inside each dt, dd, or dl tags whether as place keyword, theme keyword or stratum keyword. As we mentioned before, HTML is used for displaying data, there is no reasonable method to process the data. This is why users cannot find a document with place keyword equal to Hong Kong for example, but the users can find a document that contains the provided keyword as it is identified somewhere within the document. In Figure 2.5, this shows a browser view of the HTML document and this view is a much clearer showing the hierarchical structure of keyword element.

```
<?xml version="1.0"?>
<keywords>
        <theme>
                <themekt>None</themekt>
                <themekey>Superficial Geology</themekey>
        </theme>
        <place>
                <placekt>None</placekt>
                <placekey>SAN TIN</placekey>
                <placekey>SHEUNG SHUI</placekey>
                <placekey>KAT O CHAU (CROOKED ISLAND)</placekey>
                <placekey>CASTLE PEAK</placekey>
                <placekey>YUEN LONG</placekey>
                <placekey>SHA TIN</placekey>
                <placekey>SAI KUNG PENINSULA</placekey>
                <placekey>TUNG CHUNG</placekey>
                <placekey>SILVER MINE BAY</placekey>
                <placekey>HONG KONG ISLAND AND KOWLOON</placekey>
                <placekey>CLEAR WATER BAY</placekey>
                <placekey>SHEK PIK</placekey>
                <placekey>CHEUNG CHAU</placekey>
                <placekey>HONG KONG SOUTH AND LAMMA ISLAND</placekey>
                <placekey>WAGLAN ISLAND</placekey>
        </place>
        <stratum>
                <stratkt>None</stratkt>
                <stratkey>Superficial</stratkey>
        </stratum>
</keywords>
```

Listing 2.4 An XML document for describing the keyword element of CSDGM.

To create a XML document for describing keyword element, a number of new tags need to be introduced. To determine what tag to create is simple in this case, that is to find out all the attribute name and value pair, change the attribute name to tag name while the value is the text enclosed by the tag name. Listing 2.4 shows a XML document using markup language for keyword element.

As you can see from the Listing for describing keyword, the first line of the document is a text declaration: <?xml version='1.0'?>, showing this is a XML document and in fact for each XML document, this text is required to be placed as file header. The XML document structure is very clear and simple to read and understand. In addition, the original hierarchical structure can easily be shown without any ambiguity. From each start tag e.g. <keywords> to its end tag </keywords>, it must be a pair and there is a value enclosed in between, as a whole it is an element. With semantics and structure, it becomes a simple task to find out if there is place keyword equal to particular value and it is no longer a search for document that only contains the keyword. But more precisely, which part the keyword is contained in the document.

Furthermore, XML documents could contain multiple elements depend on the application. However, not all the elements contain data. Some elements are intentionally left empty and some elements contain attribute list. If an element is declared to be empty, it has no content and it can end with "/". For example, <keyword />. If an element contains attribute list such as a theme type list, it could define as <keyword type="theme" />. In this case, 'type="theme"' is an attribute pair, which contains the attribute name 'type' and attribute value 'theme' for the empty element 'keyword'. It is possible for an element to contain multiple attributes.

2.5.4 Well-formed XML

If a XML document is well formed, it is parsable by conformance XML processor without any notational and structural error. In fact, for XML document to be well formed, it should meet all the well-formed constraints such as:

- Single root

There is an element called root, other elements would be contained by the root i.e. start tag and end tag for the root. In addition, for all other elements, if it is a content of another element, it should be enclosed by the tag pair of the element, which contains it. In other words, there is a strictly hierarchical structure for a XML document.

- No unclosed elements except empty element

All the elements should have corresponding start tag and end tag. However, if an element declared to be empty and denoted with proper syntax, it is possible for it to ignore the end tag.

- Attribute values must be enclosed by quotes (“)

Every attribute value must have quotes, no matter the value is a text or number.

- Using character entities to replace characters such as <, >, and “

Text enclosed by tags contains characters such as less than, i.e. <, greater than, i.e. >, and double quotes, i.e. “, should be replaced by the following character entities respectively, i.e. &lt;, &gt; and &quot;.

2.5.5 Validated XML & Document Type Definition (DTD)

If one proves a XML document is well-formed, this does not mean the document is valid. A XML document is valid if it has an associated document type declaration and if the document complies with the constraints expressed it [Bray et al., 1998]. The document type declaration within a XML document possibly points or contains grammar for creating the underlying markup document. The grammar is known as document type definition (DTD) that can be a separated file or contained in the same document. For all XML processors, it should able to check XML document against its document type definition for validation. If discrepancy found, there is error and should be reported to users for correction. Listing 2.5 shows an example of a document type declaration for a XML document.

```
<?xml version="1.0"?>
<!DOCTYPE keyword SYSTEM "keyword.dtd">
<keyword>
<!— other elements go here --!>
</keyword>
```

Listing 2.5 Example of document type declaration.

The example in this Listing contains a document type declaration denoted by a string "!DOCTYPE". Following is the root element of this XML document, which is called keyword in this example. Then a system identifier that provides a universal resource locator (URL) points to a DTD file i.e. keyword.dtd.

```
<!-- This is an exmple DTD for the keyword element within CSDGM--!>
<!ELEMENT keywords (theme, place?, stratum?, temporal?)>
<!ELEMENT theme   ((themekt, themekey+)+)>
<!ELEMENT place   ((placekt, placekey+)+)>
<!ELEMENT stratum  ((stratkt, stratkey+)+)>
<!ELEMENT temporal ((tempkt, tempkey+)+)>
<!ELEMENT themekey (#PCDATA)>
<!ELEMENT themekt  (#PCDATA)>
<!ELEMENT placekey (#PCDATA)>
<!ELEMENT placekt  (#PCDATA)>
<!ELEMENT stratkey (#PCDATA)>
<!ELEMENT stratkt  (#PCDATA)>
<!ELEMENT tempkey  (#PCDATA)>
<!ELEMENT tempkt   (#PCDATA)>
```

Listing 2.6 A document type definition (DTD) for keyword element of CSDGM

Document type definition provides grammar for a markup document and it is normally defined according to a particular application. In this example, our application is to describe keyword element of CSDGM. Listing 2.6 shows the DTD that defines how to construct a document containing the keyword element.

In the Listing, the first line is comment. The other lines are statements that use to declare element type and its relation to other elements. For example: <!ELEMENT keywords (theme, place?, stratum?, temporal?)>. The syntax of element declaration is <!ELEMENT Name (content model)>, of which the "!ELEMENT" is a element declaration identifier, "Name" is a element type name, and content model describes organization of children elements such as whether it contains optional elements, a choice of elements or a sequence of elements etc. In the example, "keywords" is the element name that the statement declares. In addition, it is the outermost element or root element, which may contain multiple elements or child and is a sequence. In this example, the keyword element contains four other elements.

Between each element, there is a separator "," and all the elements should appear in order. Instead, if this is a choice, "|" would be used and only one element type would be picked from the list and appear in the document. To control the multiplicity and which element is required or not to appear in the content, there are three optional characters follow each element name to indicate this. Optional characters indicate occur of element may be one or many (+), zero or many (*) and zero or one times only (?). If the optional character is absent, the element should appear exact once. At the end of the Listing, element declaration of content model only contains keyword "#PCDATA". This indicates that the element contains parsed character data and the XML processor can only find character data inside. There are other keywords such as "EMPTY", which means that the element cannot contain anything. In addition, if there is keyword "ANY", this means the element content can be any text or other element, etc.

Beside the element declaration, there are other methods to put data in DTD. The first is the attribute declaration, which allows an attribute list with attribute name and its value appears within an element i.e. inside a tag.

```
<!ELEMENT keyword (#PCDATA)>
<!ATTLIST keyword type (Theme | Place | Stratum | Temp) "Theme" #REQUIRED>
```

Listing 2.7 An example of attribute declaration

In Listing 2.7, keyword types can be organized in an attribute list. Declaration of attribute list uses identifier "!ATTLIST" following by the related element name, attribute name, a attribute value list, a default value and an operation keyword which specify if the attribute is optional i.e. #IMPLIED or is required i.e. #REQUIRED.

Another method to put data in DTD is entity declaration, which is used to create a symbol to replace a series of text. For example in Listing 2.8, app is a symbol for text "Application". When the symbol is used in a document such as "This is an &app;", it will be rendered as "This is an Application".

```
<!ENTITY app "Application">
```

Listing 2.8 An example of entity declaration.

In fact, with all these clear and simple DTD features, it tells how a XML is structured and enables to valid corresponding XML document.

2.5.6 Document Object Model (DOM)

Document Object Model is a specification of applications programming interface (API) standard currently developing by the W3C [Apparao et al., 1998]. The DOM is composed of two levels. The first level is called DOM Level 1, which is completed in October 1998 and now a W3C recommendation. This specification is concentrated on the development of document model, which provide definitions for manipulating and navigating document objects and it is a core of the entire specification. While second level is still under development started in late 1998, which is called DOM Level 2. This level includes various supporting standard for XML such as style sheet object model, which provide formatting for the document, functionality for manipulating the style information that attached to a document, support for XML namespaces and so on. In this project, we will focus on DOM Level 1 only, since the functions provided would not be used within the project scope and in fact, the DOM Level 2 specification is not stable yet. Nevertheless, some of the software vendors such as IBM have already developed XML processor based on DOM Level 1 specification.

Traditionally, information is stored as data using various systems. In DOM, it stores information as a document and defines a logical structure for document as well as the ways to access and manipulate it. Typical functions such as build a document, navigate its structure, add, modify and delete element and content can be found from the DOM. In addition, to make the DOM a standard API that can be used by various applications, it provides binding to different programming languages. Available binding in DOM including JAVA, which is an object-oriented language, and

ECMAScript, which is an industry-standard scripting language based on JavaScript and JScript.

Consider the "Keyword" element in the previous example, it can be represented in DOM as shown in Figure 2.6. In the Figure, "Keyword" resembles a tree that contains a single root and multiple nodes. From the view of DOM, each node, no matter it is a root or not, is an object, which has functions and identity.

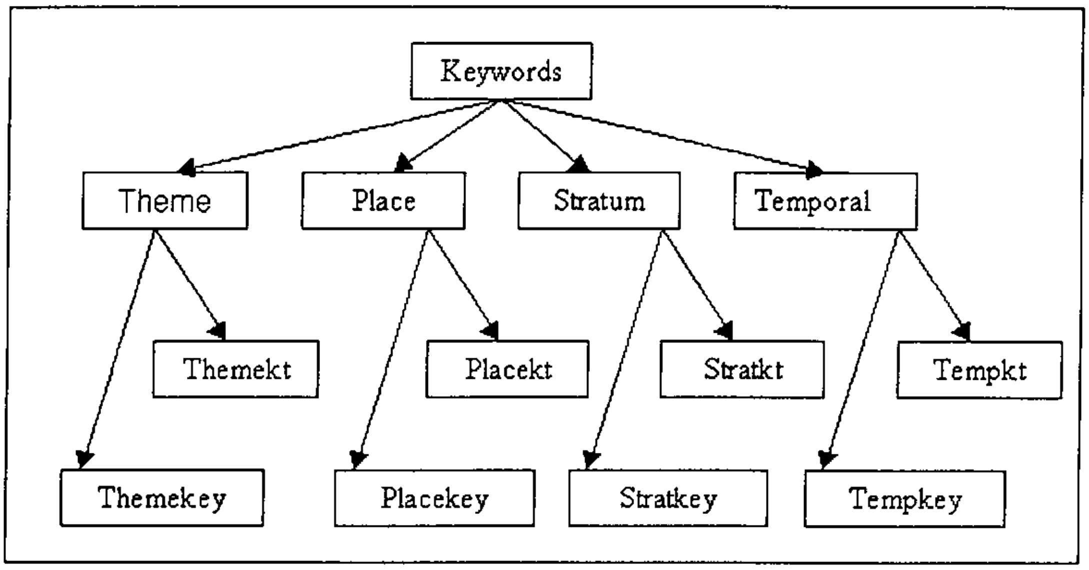


Figure 2.6 An example shows the document structure for describing the keyword element of CSDGM.

In this section, we have introduced the development of XML and its mechanism to create documents for describing information. In addition, the implementation of DOM helps to manipulate XML document and enables development of various XML applications. The only limit is the user's imagination. In the following section, we will start to explore opportunities to use XML, DOM for managing geospatial application information.

2.5.7 Potential applications with the use of XML, DOM for managing Geospatial Application Information

Currently, the CSDGM provides guidelines for preparing geospatial data level of metadata. There are no related guidelines for describing geospatial application, which provides understanding about how GIS procedure providing solution to specified geospatial analysis and modeling problem. It is expected that the description for application should play the similar roles as in CSDGM for building metadata such as facilitate analysis procedure reusability, its sharing and investment protection on data creation and maintenance. Ultimately, a set of application description could provide a foundation for building an application base management system, which provides different kind of data management service for problem-solving technique related to geospatial analysis and modeling problems. It is expected that the application base management system for GIS should provide two levels of services. The first level or primary service provides typical data management facilities for users to store, retrieve and view application knowledge. The second level service provides advanced capabilities such as interface to geospatial data clearinghouse for searching and retrieving CSDGM compliant metadata documents, provide ready-to-run geospatial analysis and modeling procedure for specify GIS software, and capability to translate commands between different GIS software. To enable the role of metadata plays, it is necessary to allow processing of the application level metadata, interacting with other applications i.e. GIS, displaying information according to users' requirements and accessing from a large amount of users. Accordingly, the XML and DOM technology could help to build the application base information system. It is because XML allows building data definition and semantic in term of document, this in fact solves a major problem that produces self-describing data. Meanwhile, it is not working in binary coding like Z39.50, which solely uses simple text that allows users to read and edit the content without special requirements. It is also designed to be Internet enabled which could integrate with other web technologies such as HTTP, HTML, and multi-language support, which allows XML document written in different language but carrying the same contents by using the same document type definition. In addition, with the DOM technology, it enables processing of XML documents. Since DOM is in fact a specification for building XML processor, a number of implementations are available in the public domain for development, hence it allows software developers concentrate on the specify application rather than the basic data manipulation.

On the other hand, the Z39.50 searching and retrieving protocol that was chosen to build geospatial data clearinghouse could be used to build the proposed application base management system. However, the technology is not widely accepted in the Internet and it is originally designed for library cataloguing used only. It is not working in ASCII as basis, which generate difficulty to implement, i.e. involves encoding and decoding of data into and from binary code whenever the data is being retrieved from or put back to the server. If users want to communicate using this protocol, they are required to use special software, although there are adds-on for Internet browsers, which enable it to communicate using the protocol, it is not convenient to do so. As we can see, the Z39.50 developers also aware the HTTP and HTML is the most popular combination in the Internet, they would like to enable Z39.50 information available for the HTTP users. As a result, the developers create a Z39.50-HTTP gateway that resides on the Z39.50 server and allows HTTP messages map to Z39.50 messages and hence it could greatly increase the usability of the facility. Nevertheless, the main advantage of Z39.50 is its stateful connection that allows users to maintain a session within a transaction period. To start a transaction, users are required to provide instruction to do so. To end a transaction, users are also required to log off the server or the server will log out the users automatically due to time out. This provides capability for servers to trace client activities and for users to refine query set and produces precise results to what the users want to find. Instead, in HTTP, there is a mechanism to make the original stateless connection work as stateful that it uses a small set of data called cookie resides on the client machine that provides information and help the server to know the connection status. Nevertheless, whether the connection type between server and client is stateful or not is insignificant in our project. Our concentration is to find the best solution to describe related geospatial resources such as geospatial data that have been done by the CSDGM and related works, and application information that help users to understand geospatial analysis approach to particular geospatial problem and do actual analysis using GIS. As a result, we would choose to use XML and DOM as a basic technology to handle application-related information. Base on this technology, we can develop architecture to build system fit to the application base information system level 1 and level 2 service respectively.

The level 1 service requires defining structure and semantics of geospatial application information. This is similar to define a schema for a database application. Once the structure and semantics are defined, it can be used to produce and validate XML document. Base on this, a simple information browser that allows data rendering, browsing and querying could be developed. The document data could be handled by different database systems depending on the conditions. If there is limited resources, the size of data set is small and requires less coding effort, it may use file system to handle the XML document. On the other hand, if there are plenty of resources, large data set and require comprehensive data management capabilities, it may use database management systems to handle the XML document. In addition, DOM would handle manipulation of the document data, while any programming language may be used to develop the user interface. Following this approach, there are two architectures to build the level 1 service. The first architecture is to build a data rendering and manipulation application on top of XML processor. This simple standalone program provides limited functions to handle data. However, it can help to prove the concept of building application level metadata and its usage. Another architecture is more realistic and practical, since it is composed of a web server, client and XML processor. It is expected the functionality that this architecture can deliver is the same as the first one, however it enables better and more robust data distribution and sharing in an Internet environment. Basically, the data flow involves several steps as follows:

1. Client browser submits a request using HTTP with HTML to the web server,
2. The web server returns XML document to a XML application according to users' request,
3. The XML application processes the XML document and retrieve useful information from it,

4. The XML application then formats the extracted information to HTML and send it back to the client browser and,

5. The client browser receives the HTML document and displays it properly with useful information.

This should be the most typical architecture to build XML applications. The significant part of it is the XML document and the processor that can response to client requests accordingly. In most cases, this application should be specified such that it is designed for particular kind of data such as geospatial data.

The level 2 service is much more complicated and involves several add-on modules to the level 1 architecture for manipulating XML document and interacting with other systems such as, database system, web server, interface to geospatial data clearinghouse and interface to GIS. The database system is not essential, but it definitely can provide comprehensive data management that meets the requirement to provide a foundation for data searching and maintenance. Beside the web server and the XML application similar to level 1 architecture, there is an interface to geospatial data clearinghouse that uses Z39.50 protocol to do search and return result to the user. Moreover, the application management module should provide parameters editing and procedural editing capability to users, hence they can edit the retrieved application procedure and modify required parameters or combine various procedure to form a new one according to their own application. If the procedure is validated and the connection to particular GIS is available, the application management module should be able to submit the processing request to the GIS. Since the application information may contain different GIS processing procedures doing the same task, it might be used to map GIS commands between various GIS software. Hence, it provides information on how to do command translation among different software, which help users to learn new software much easier.

To sum up, XML and DOM could be used to build foundation for describing and processing geospatial application. However, it is the implementation issue, prior to this is to build a data model that can appropriately describe the application information. In the following chapter, we will provide the development of the data model for geospatial application, which is called application model.

CHATPER 3
THE APPLICATION MODEL

3.1 Conceptual Model

The motivation of this project is to help GIS Analysts or users to find appropriate geoprocessing procedures and related spatial resources for solving spatial analysis and modeling problems. To accomplish this, our approach is to develop a well-defined description of geoprocessing procedure as a foundation for documentation. This description is developed based on an application model. With the structure provided by the application model, we can develop applications such as geospatial application information browser or ultimately, a geospatial application knowledge base. This chapter will describe the analysis and design of the application model and the its extension to the development of geospatial application information browser. In addition, we choose to use object modeling technique (OMT) methodology [Rumbaugh et al., 1991] for generating and constructing object models of the application model and the geospatial application information browser throughout this study. The entire set of OMT notation for object model, dynamic model and functional model can be found in the Appendix II of this thesis.

As shown in Figure 3.1, the objective of the geospatial application information browser is to allow browsing of application descriptions from an application description store, checking description error, searching and retrieving metadata document from geospatial data clearinghouse database.

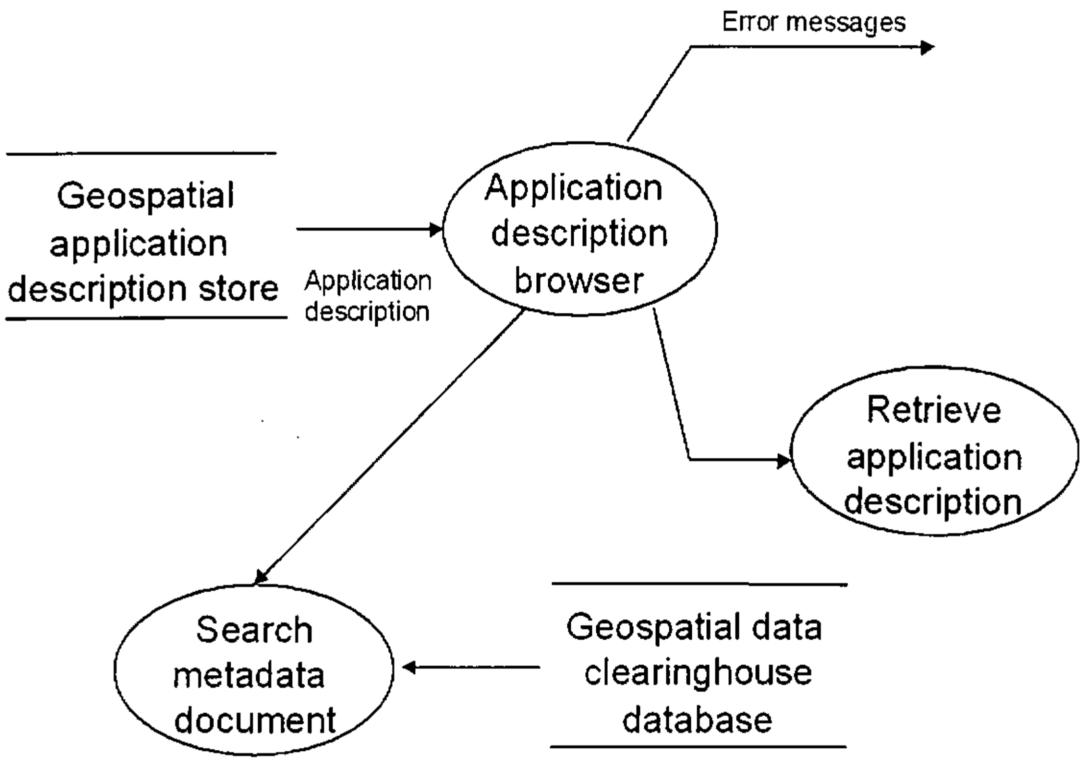

Error messages
Geospatial application description store
Application description
Application description browser
Retrieve application description
Search metadata document
Geospatial data clearinghouse database

Figure 3.1 Functional model of the geospatial application information browser.

We choose to develop an information browser because it is simple and provides rapid prototype for applying the application model. It is common that GIS users are unaware of document geoprocessing procedure, which leads to difficulties for sharing geospatial problem solving technique among the user network. In addition, this also disables the reuse and trace back of the procedure. Although the geospatial metadata standard such as CSDGM contains data items for lineage information, it is not precise enough to provide descriptive information for understanding the processing procedure. The application model helps to fill this gap. Similar to the geospatial metadata standard, GIS users can follow the model structure and create description for documenting geospatial applications. The information browser provides primary functions for browsing application descriptions easily and enabling information sharing. Ultimately, by collecting multiple geospatial application documents with database technology, it could build a knowledge base dedicated for geoprocessing. As a result, no matter GIS users are novices or experienced, they can retrieve information from the database and find ready to run application procedure easily via a simple user graphic interface.

In the following sections, we will focus on the analysis and design of the application model. Then, based on the application model, we will design a geospatial application information browser for viewing the model and helping users to find geospatial application resources.

3.2 Analysis of Geospatial Application

A geospatial application is a sequence and combination of operations. It is used to instruct Geographic Information System (GIS) and provide solutions for solving particular geospatial analysis or modeling problem. For example, consider the following geospatial problem: Finding affected parcels if the gasoline tanks in all gas stations leak at the same time. No matter to which area for the study, we would provide the following pseudo code to model the scenario.

1. Retrieve from database all gas stations.
2. Generate buffer zones around all the gas stations to simulate the area affected by a leakage.
3. Overlay the zones and the parcels to find the affected parcels.

Although the above pseudo code cannot provide the final answer, this provides a concept for solving the problem. In addition, this is also documentation, which is independent to any data processing system, for users to determine which operations and data would be applied for later implementation. In order to give a realistic answer to the problem, only pseudo code is not sufficient, processing information for particular GIS is required for implementing it. However, GIS itself does not understand pseudo code and we need to translate it to system understandable operations. Although different geographic information system would provide possibly the same functions for operations, their syntax and parameters used are different. To provide a workable set of operations, this requires a mapping between the pseudo code and a sequence of GIS commands.

To construct a workable set of GIS commands using the pseudo code, we choose CARIS for Windows [Lee, 1995] from Universal System Limited for providing a sequence of operations, which implements it and provide qualitative answer to the problem.

1. Open CARIS Information Manager. A panel with title "Enter name of CARIS file to open" would be opened.
2. Enter file name: A spatial data file for an area. Click Okay to close the panel.
3. Select Query>Query Condition, a menu item from the Database Manager window. A panel with title "Query Condition" would be opened.
4. Enter data value for each variable in the panel accordingly such as Name: Property, Column: PROP_TYPE, Operator: = and Value: 203. Then click Add and OK to close the panel.

5. Select CARIS Information Manager>Options>Resolution. A panel with title "Pixel Resolution Settings" would be opened.

6. Enter value for the user-defined resolution value i.e. 5. Click Okay to close the panel.

7. Select CARIS Information Manager>Zone>Create>By Corridor. A panel with title "Zone Corridor Settings" would be opened.

8. Enter value for each variable in the panel such as Buffer Width in Ground Units: 200, Zone Name: ZONE_A and select Current Selections to generate buffer for current selection. Click Okay to close the panel.

9. Open CARIS Information Manager>MapQuery>By Zone and a panel with title "Map Query by Zone" would be opened.

10. Select appropriate value from the panel such as choose ZONE_A from the Choose Zone box and choose Polygon from the Choose Feature Type box. Then click Okay to close the panel and execute the operation.

As a result, a group of parcels within 200 meters of any gas station would be shaded and displayed on screen, and this provides the solution to the question: Finding affected parcels if the gasoline tanks in all gas stations leak at the same time. The GIS operations to the problem involves a series of steps, especially when we describe operations using graphical user interface because this involves multiple value entry and selection actions.

On the other hand, if we choose to use another GIS for implementing the same operation, it would generate another set of commands. In this case, we use Modular

GIS Environment (MGE) from Intergraph [Intergraph, 1994]; the following commands provide solution to the same spatial analysis problem.

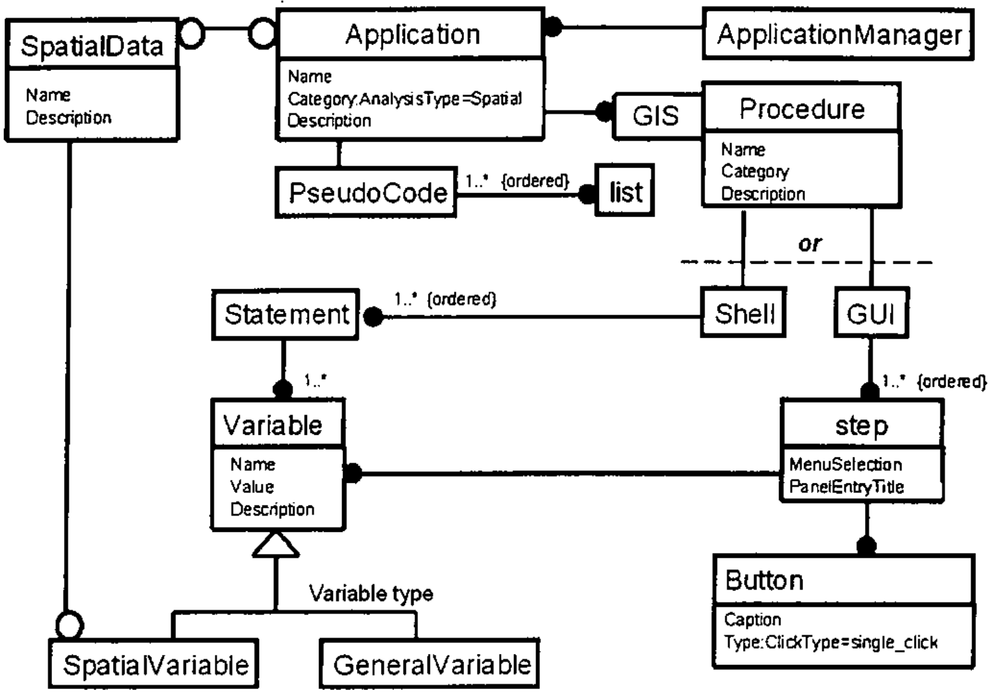
1. ulfbldr -v -I fton.dgn -E parcel.ulf -l 10 11
2. tpdbr –v –A parcel.ulf -O parcel.top
3. qybdr –v –T parcel.top –I query1 –q "elements from parcel where PROP_TYPE=203;"
4. tpzonr –v –T parcel.top –z "200 m" –O zone_A.top –I query1 –a
5. tpdbr –v –T parcel.top –T zone_A.top –O result.top
6. qybdr –v –T result.top –I query2 –q "elements from parcel overlaps elements from zone_A
7. dgnbdr –v -T result.top –D result.dgn –I query2

The implementation gives a clearer operation pattern that provides instruction for GIS to process the resulting files i.e. result.top, result.dgn, for presentation. In addition, each line is a statement. A statement is a representation of operation with well-defined syntax [Tomlin, 1990]. Nevertheless, no matter the operation is represented by steps or statements, each of it contains three types of information. Firstly, there is an operation or command, which determines the processing action. For example, to open a panel such as "Select Query>Query Condition" in the first case or the key-in command such as "ulfbldr" in the second case. Secondly, there are modifier names within panel or a key-in command. Thirdly, for each modifier, a value would be assigned to the modifier. In the case of using command line, the value is assigned after the modifier i.e. value "parcel.top" follows modifier "–T", which tells the system, the input topology file is "parcel.top". Since this value can be changed according to different situation, it is a variable. This variable can be classified into two types. The first type is used for supporting uses that provide options for processing data. The second type is data file. Typically, this is a spatial data file name for input, intermediate use or output. However, the data file may include of other type such as text file or word document.

Using the model, a geospatial application can be described by a series of ordered statements or steps, which forms procedure for solving particular geospatial problem. In addition, a statement composes of operation, modifiers and its values. However, a geospatial application can only provide a series of pseudo code for describing the idea of solution to the problem, this is only a model for solving the spatial problem. Therefore, we need to identify a geographic information system and provide appropriate sequence and combination of commands, modifiers and spatial data match to the system in order to generate a realistic solution.

3.2.1 Object Model for Geospatial Application and Geospatial Data

The object model of geospatial application is shown in Figure 3.2. Base on the analysis of geospatial application, the developed object model helps us to understand and find appropriates procedure and data sets for performing GIS applications. The description should include pseudo code and operational procedures. The pseudo code provides understandable description for users to solve spatial analysis and modeling problem. It must be GIS independent which means any GIS can make use of it to implement and produce a real solution to the spatial problem. Besides, the operational procedure is an implementation of the pseudo code and is GIS software dependent, which comprises ordered statements containing related GIS commands and variables. There are two types of GIS interface including graphics user interface (GUI) and command shell.



{Application must be acyclic}
{AnalysisType includes Spatial, Grid, Network, Terrain, Image, Mixed}
{ClickType includes single_click, double_click, and select}

Figure 3.2 Object model of the geospatial application

Currently, most GIS provide intuitive GUI for users to process geospatial data. For batch processing, some GIS would use command shell to do processing. In the object model for geospatial application, it enables the description of both types using different components such as statements for command shell, steps for GUI and so on. In addition, no matter which interface the GIS use, there are variables in between each statement or step to process. There are two types of variables that are described in the model. They include general variable and spatial variable. Firstly, general variable describes command modifiers or its supporting value such as distance. Secondly, spatial variable describes data item that is a spatial resource such as GIS coverage.

For each spatial variable, it is also self-describing and it could provide information to help find spatial data. Four classes of keywords are used to describe the expected spatial data including themekeyword, placekeyword, tempkeyword, and stratkeyword. It is expected to use these keywords for identifying and finding metadata documents that are compliant with CSDGM. In fact, this keyword set is borrowed from the keyword section of CSDGM. With the additional information such as the data name and its description, it forms SpatialData object as shown in Figure 3.3.

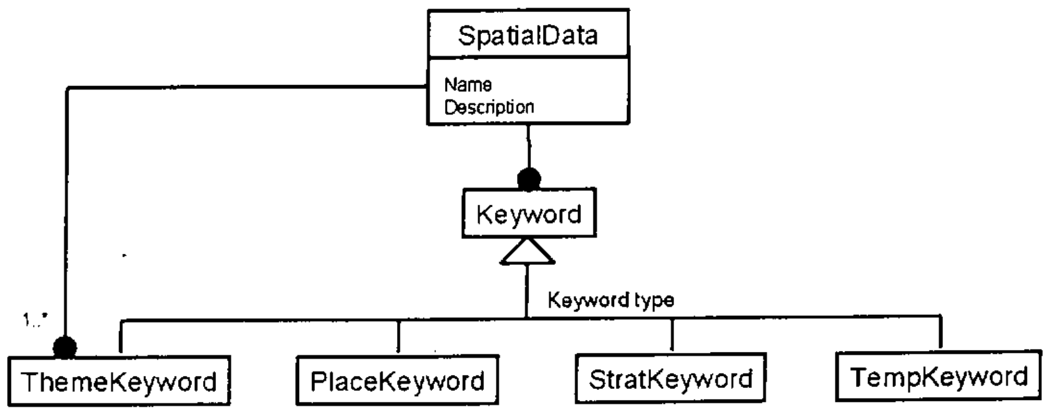


Figure 3.3 Object model of spatial data

Each SpatialData object could be used for describing the output data as a component of particular Application object. In this case, an application hierarchy could be formed. This hierarchy is a directed graph structure and this should be maintained. Otherwise, cycle would be formed and it will generate an infinite loop when traversing the object relationship between the SpatialData object and the Application object. For example, if a combination of application A and dataset A produces dataset B, and a combination of application B and dataset B produces dataset A, this forms an infinite loop when traversing data set A for finding its originate. This is logically impossible, since information contents contain in source data e.g. data set A should be higher than that of its derivative, e.g. data set B. This should be taken into account during implementation; otherwise, it would create a deadlock during querying.

Furthermore, SpatialData object is part of the Application Model and it will not be used independently.

3.2.2 Data Dictionary

This section provides definitions for the objects of the Application model and SpatialData model respectively.

Application – It provides both the concept and its implementation that applies geoprocessing technique for solving spatial analysis and modeling problems. There are different types of geospatial applications including spatial analysis, network analysis, image analysis, grid analysis and mixed analysis. The mixed analysis type means the processing required more than one typical geo-processing type to perform e.g. spatial analysis and network analysis. In addition, this object contains two other attributes, which is the application name and its description.

ApplicationManager – It is in fact not part of the object model of the geospatial application. It is instead an object for managing geospatial application objects. It provides interface between the application object and users as well as interaction between machine via network for data retrieving and searching uses.

GeneralVariable – It is a type of variable that could be a modifier or a value in a geoprocessing command within a statement or a GUI panel. For example, a spatial selection command may require a value naming 'd', which tells the command to create a buffer zone about a point at d meter. In addition, the command may provide a modifier naming 'inside', which tell the command to perform selection within the buffer zone.

Procedure - It is a set of ordered statements or steps related to GIS software. It is used to implement the spatial analysis and modeling concepts describe in PseudoCode. In addition, it is qualified by known GIS software.

PseudoCode – It is a documentation or description that helps users to understand the concept of solution. It is system independent and for later implementation uses.

SpatialVariable - It is a type of variables that are spatial coverages denotes within a statement or a GUI panel.

Statement – It is an expression contains well-defined syntax, commands, variable names and its values. Multiple ordered statements form procedure for performing a spatial analysis or modeling application.

Variable – It contains a pair of parameters represent the variable logical name and its value. The value is varying according to different criteria. Criteria like defined distance for creating a buffer zone, command modifier for identifying elements in particular data layer.

GUI – Graphical Users Interface (GUI) provides interaction between users and system for data input and operation. It is one of the two methods for describing implementation of the PseudoCode.

Shell – It is a user interface for entering commands. By providing key in command with its modifiers, the command would be executed for data processing. It is another method for describing implementation of the PseudoCode.

Step - It is operational sequences of GUI operation, which describe user interaction with GUI such as button clicking, selection and so on.

Button – It is a button required clicking or selecting within a graphic panel when GUI is in used.

The following is the definition of objects for SpatialData as shown in Figure 3.3:

SpatialData - A data set describes in term of a set of keywords. It could serve as input within a procedure. It could be linked to an Application, hence it can determine the data set that generates by it.

Keyword – words or phrases, which is used for summarizing an aspect of the spatial data set. It is composed of four keyword types: theme, stratum, place and time.

ThemeKeyword – Word or phrases, which is used for describing the subject of the spatial data set.

StratKeyword – The name of vertical location used to describe the locations covered by a data set, e.g. Sub surface, seabed and so on.

PlaceKeyword - The geographic name of a location covered by a data set.

TempKeyword – The name of a time period covered by a data set.

3.2.3 Dynamic Model

Dynamic model is used for describing interaction between model objects and users over time. Basically, it helps to define the user interface. Most of the objects in the proposed object model are static, which provides structure for implementing a geospatial application database.

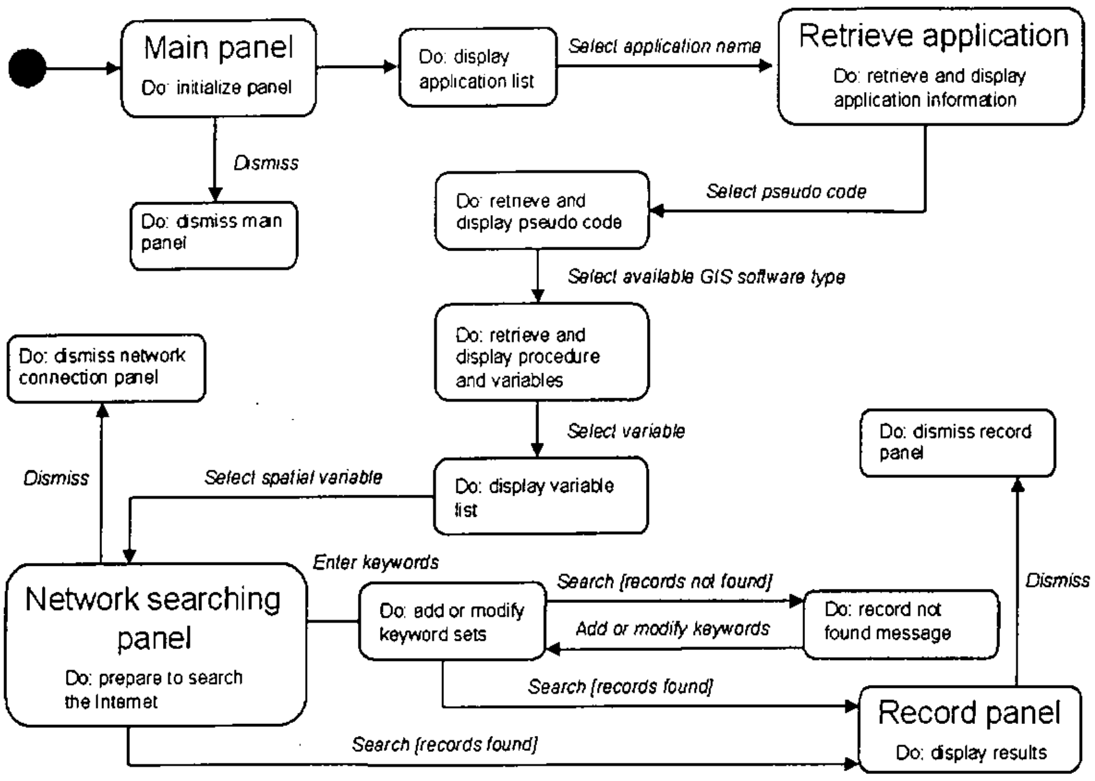

Main panel
Do: initialize panel
Do: display application list
Select application name
Retrieve application
Do: retrieve and display application information
Dismiss
Do: dismiss main panel
Select pseudo code
Do: retrieve and display pseudo code
Select available GIS software type
Do: retrieve and display procedure and variables
Do: dismiss network connection panel
Select variable
Do: dismiss record panel
Dismiss
Select spatial variable
Do: display variable list
Enter keywords
Network searching panel
Do: prepare to search the Internet
Do: add or modify keyword sets
Search [records not found]
Add or modify keywords
Do: record not found message
Dismiss
Search [records found]
Record panel
Do: display results
Search [records found]

Figure 3.4 State diagram for the ApplicationManager object

Nevertheless, in the proposed model, ApplicationManager object is responsible for serving other objects and handling user interactions. The interaction in this system is typical that involves object querying, and data rendering operations. We would write down possible interactions between user and the ApplicationManager. Meanwhile, Figure 3.4 shows the state diagram of the ApplicationManager, which illustrate different user interaction during the ApplicationManager in operation.

In addition, the following is a typical scenario for showing the interaction between users and the ApplicationManager object in the proposed model:

1. ApplicationManager reads descriptions from geospatial application store.

2. ApplicationManager provides a list of available geospatial application names.

3. User selects one of the application names from the list.

4. ApplicationManager retrieves application information such as variables, pseudo code, available GIS software scripts to users according to the selected application name.

5. User selects one of the spatial variables available from the variable list.

6. ApplicationManager retrieves related spatial data description of the selected spatial variable.

7. ApplicationManager starts the network-searching panel with the spatial data description of the selected spatial variable.

8. The network-searching panel asks for spatial metadata clearinghouse server name, database name, and IP address. The user enters required parameters and connects to the server.

9. ApplicationManager provides a set of pre-defined keywords in four themes such as theme keywords and place keywords for metadata searching. User can also add or modified the keyword set.

10. ApplicationManager waits for the user to start the search.

11. User starts the search. The ApplicationManager then creates and sends a query to the server. If the record is found, ApplicatonManager will retrieve the record and open a record panel to display the record for the user. Otherwise, a record not found message would be popped up.

3.2.4 Functional Model

The functional model defines operations that objects perform. The geospatial application description developed based on the application model is readable by computer and users easily. Properly, it would be written in ASCII format. For each time it is read by the computer, it parses the description and builds model according to pre-defined object model definition. Once the application object model is built in memory, the information browser will further render the model using different UI components such as tree view, table view and combo box.

The geospatial application description store will serve as storage for all available application description. On the other hand, there is another data store for geospatial data description. This geospatial data description is designed for determining metadata document with a set of keywords. It is part of the geospatial application description. In addition, the contents of geospatial data description is to provide useful and meaningful keywords for searching the geospatial data clearinghouse database without guesswork. Figure 3.5 elaborates the functional model of the geospatial application information browser.

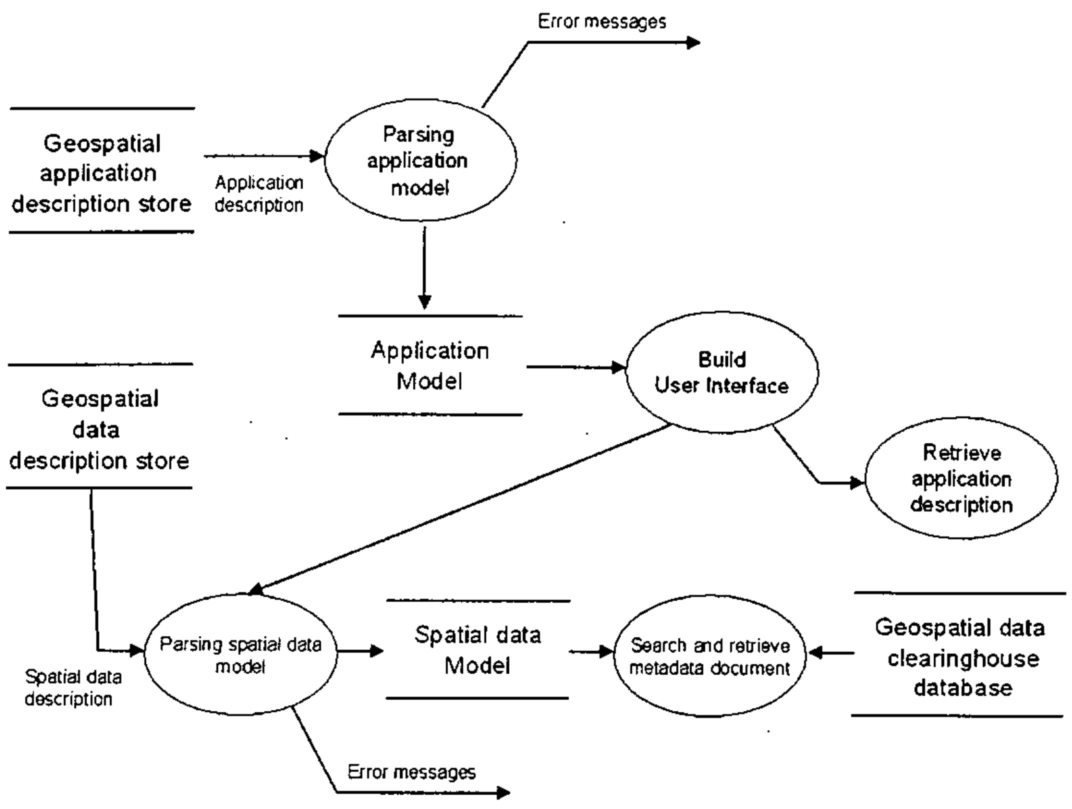

Error messages
Geospatial application description store
Application description
Parsing application model
Application Model
Build User Interface
Geospatial data description store
Retrieve application description
Spatial data description
Parsing spatial data model
Spatial data Model
Search and retrieve metadata document
Geospatial data clearinghouse database
Error messages

Figure 3.5 Elaborated functional model of the geospatial application information browser

3.2.4.1 List of Identified Functions

The following are all identified functions provided by objects in the proposed object model.

- Add, delete and update an application.
- Add, delete and update a pseudo code of an application.
- Add, delete and update a procedure regarding a particular GIS.
- Add, delete and update operational steps for manipulating GUI.
- Add, delete and update operational statement for command line procedure.

- Add, delete and update a link between spatial data and particular spatial variable.
- Retrieve geospatial application or geospatial data information from the description store.
- Find an application by application name.
- Find all available procedures for an application.
- Find all variables within a procedure.
- Find metadata document using keywords in a particular spatial data.

The first six operations are typical data manipulation functions in data management such as insertion of data object, deletion of data object and modification of data object. To implement these operations using database system, it can be done by standard data manipulation language such as structured query language in RDBMS or standard object manipulation functions in OODBMS. On the other hand, if file system is used, a standard text editor is sufficient to perform all the tasks. For the remaining cases, the following is pseudocode for performing the operations.

- Retrieve geospatial application or geospatial data information from the description store.

  Application::getDocumentRoot(root name). Return a root element of a document that retrieve from the description store by given a root name. Root name could be a table name in a database system or a pre-defined identifier in a formatted file. Root element is an entry point for accessing its information contents such as geospatial application information and spatial data information. It is used for later operations that allows object traversing among underlying components or data items. Once the root element is successfully retrieved, related data item and its value will be retrieved and used to build object model for describing geospatial application or spatial data. The following Listing 3.1 is pseudocode for Application::getDocumentRoot:

```
Application::getDocumentRoot(root name) returns root element
  For each root element in description store
      If root element name == root name;
              Return root element;
      End if
  End for each
Return null;
```

Listing 3.1 Pseudocode for Application::getDocumentRoot

- Find an application by application name.

Application::findApplication(Application Name). Find the application element by a given application name.

```
Application::findApplication(Application Name)
 For each Application in Application description store
  If Application.Name == Application Name;
      Return Application;
  End if
End for each
Return null;
```

Listing 3.2 Pseudocode for Application::findApplication

- Find all available procedures for an application.

Application::findAllProcedure(Application). Find all the procedures for a given application. Starting with a root element of the given application, we traverse the children and parent association to find the relevant procedures.

```
Application::findAllProcedure(Application) returns set of procedure elements
 Set procedures;
  For each application.children
   If application.children is an instance of procedure
           Procedures.add(application.child)
   End if
 End for each
Return procedures.
```

Listing 3.3 Pseudocode for Application:: findAllProcedure

- Find all variables within a procedure.

  Application::findAllVariables(procedure). Find all the variables for a given procedure. Starting with a given procedure object, we traverse the parent and child association to find the relevant variables.

```
Application::findAllVariables(procedure) returns set of variable elements
  Set Variables;
   For each procedure.children
       If procedure.children is an instance of variable
         Variables.add(procedure.child)
       End if
    End for each
Return procedures.
```

Listing 3.4 Pseudocode for Application:: findAllVariables

- Find metadata document using keywords in particular spatial data.

  Application::Searching(keywords). Using the given keyword set to create query and send to a geospatial data clearinghouse or a Z39.50 server for metadata documents.

```
Application::Searching(keywords) returns Record
  String Query = null;
    Query = QueryBuilder(keywords)
    If NetworkConnectionOkay is true
       Search(Query)
          If RecordFound is true
```

```
                Record = PresentRecord();
            Else Message 'Records not found!'
            End if
        End if
    Return Record.
```

Listing 3.5 Pseudocode for Application::Searching

3.3 System Design

In this section, we will make use of the analysis results and move forward to determine a method for implementing the developed model and building a system for browsing geospatial application information. In addition, the system design of the geospatial application information browser is based on the following criteria:

- Ease of use - we want to provide an intuitive user interface that allows user to browse application information easily.
- Development effort - we want to build a prototype rapidly and minimize coding effort by reusing existing resources.

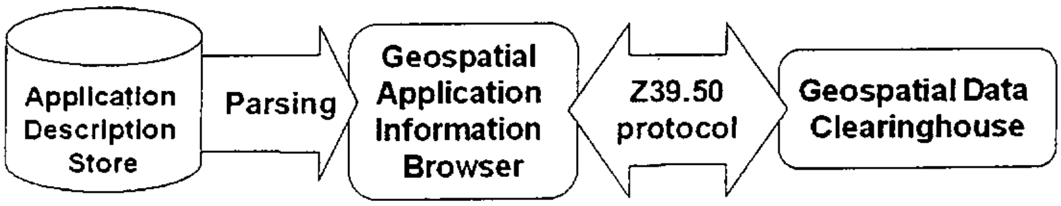

Application Description Store
Parsing
Geospatial Application Information Browser
Z39.50 protocol
Geospatial Data Clearinghouse

Figure 3.6 System overview of the geospatial application information browser

Figure 3.6 shows a system overview of the geospatial application information browser. In general, all the description such as application or data set will reside in the application description store. The geospatial application information browser will parse the description and form model in memory based on user interaction. If there is any spatial data is identified and the users would like to find further information about it. The browser allows searching the metadata document for a geospatial data clearinghouse. This is the whole mechanism of the browser.

3.3.1 System Architecture

The system architecture of the geospatial application information browser consists of three packages and two types of data files. A package is a group of objects with common theme. The three packages includes application, application.browser and application.z39. The are two types of data file forming the description store. The first type is called document type definition (DTD) files and there are two DTDs in the system, which contain production rules for both geospatial application information and geospatial data information respectively. The second type is called XML files and there are two XMLs, which contain contents based on the two DTDs. Beside the application information browsing capability, the system provides searching capability when it connects to a spatial data clearinghouse with Z39.50 searching and retrieving protocol. In addition, Figure 3.7 shows the system architecture of the geospatial application information browser.

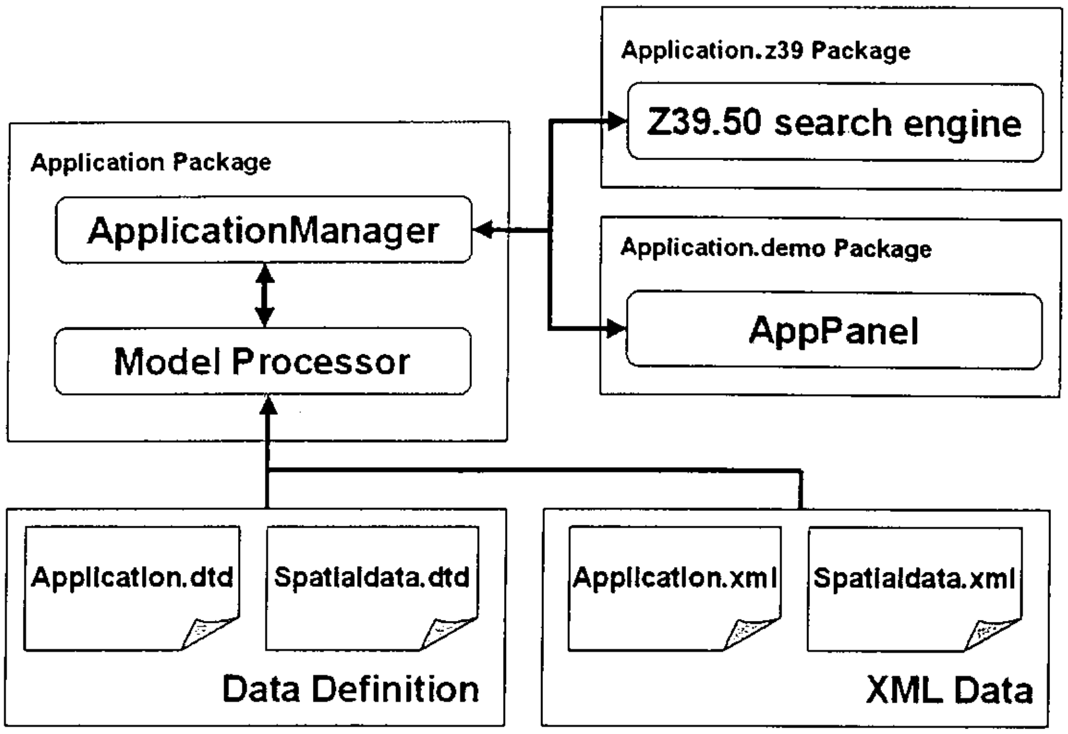


Figure 3.7 System Architecture of the geospatial application information browser

3.3.1.1 The Application Package

The application package contains two main classes including the ApplicationManager class and ModelProcesser class. ApplicationManager is an implementation of ApplicationManager object, which is mentioned in the previous object model. Its primary responsibility is to control the user interface objects in the Application.browser package. Besides, there is a ModelProcessor class that is a XML parser, which is responsible for reading and parsing data items from a XML document with the help of the corresponding DTD. After the ModelProcessor successfully parses the document, it then builds corresponding object model according to the type of data files read. The ModelProcessor can handle two file types including application.xml and spatialdata.xml. Besides, this package is also responsible for logging into a data clearinghouse server, creating and submitting queries and presenting query results. The ApplicationManager must be able to reform these to retrieve metadata document from the server by using standard Z39.50 protocol as specified in ANSI/NISO Z39.50-1995. To accomplish this, ApplicationManager delegates all the network connection works to an Application.z39 package. Once the search returns query results, it will then return to the ApplicationManager and the record will then be rendered in an appropriate GUI object.

3.3.1.2 The Application.browser package

This package contains GUI classes to build the interface of the information browser. All the user interactions are pre-defined in the dynamic model in the previous section. And this helps to decide data flow between different GUI components. In the development of information browser, we used Swing objects to build the GUI. Swing is a GUI component kit developed by Sun. Swing is part of the Java Foundation Classes, or JFC, in which it provides a list of GUI components such as panel, list box, table, combo box and so on. All the components in this package are closely integrated to the ApplicationManager class for data rendering.

3.3.1.3 The Application.z39 package

This package is dedicated to perform Z39.50 network connection. Basically, classes inside this package allows ApplicationManager to log on to preferred geospatial data clearinghouse server, create query with standard syntax, send query to the server, retrieve found records from the server and so on. In addition, this package is using a Z39.50 API developed by the OCLC.

3.3.2 Data Management

Figure 3.8 shows the data management strategy of the information browser. The Application description store is a database of the information browser. It is used to manage both application and spatial data description that follows structure of developed application model as well as the spatial data model. Ideally, we would like to use database system such as RDBMS or OODBMS to implement the application store, since database system can provide robust indexing and searching, better data integrity and standard query language.

However, the implementation procedure and technique is not trivial and simple enough to build it rapidly in this project. On the other hand, we choose to use file system for managing the data. The main reason for this is because file is easier to handle, it is delivered with the underlying operating system with no extra cost and various file editing tools are available for editing the data in ASCII format. In addition, many development tools are available for parsing and read text file from the public domain.

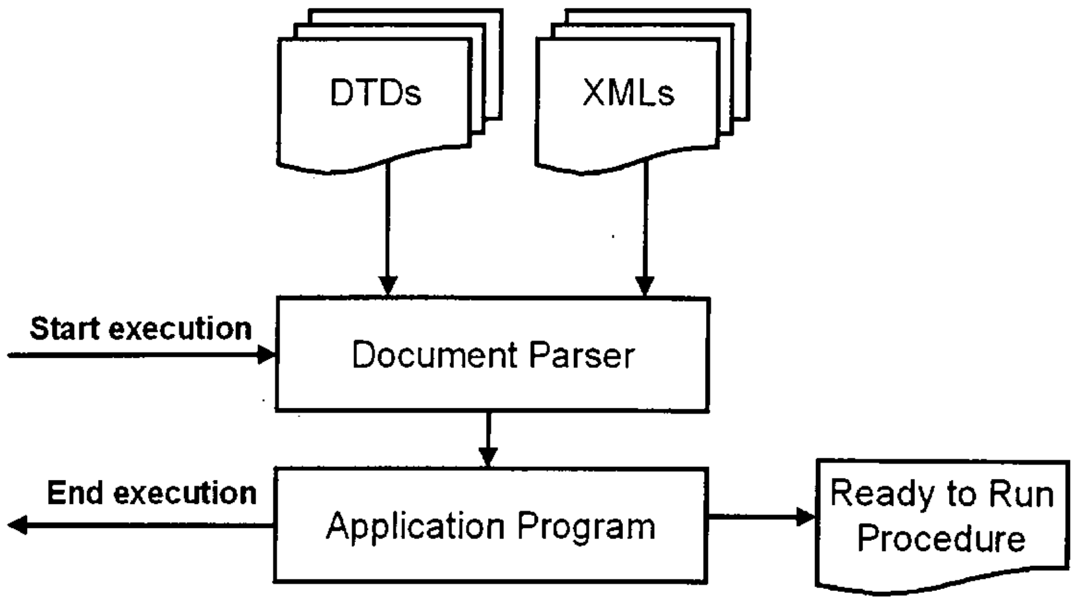


Figure 3.8 Data management strategy for the geospatial application browser

The reason for using ASCII for writing the description is because ASCII text can be read and understood by humans and its coding is simple.

As shown in Figure 3.8, it introduces a pair of documents to the system. These documents are in fact ASCII files that help build the object model in the browser. To make the file readable by the program, it uses a descriptive syntax that maps to the object model. The syntax allows carrying fruitful semantics for organizing information such as model structure and structural constraints. In this project, we would not develop a new syntax or grammar to capture the information. This is not our focus. Instead, we would use extensible markup language (XML), an industry standard markup language to develop both file definition and data. As we have mentioned in the previous chapter, XML is a new generation markup language that is similar to, but more powerful than, the commonly used HyperText markup language (HTML). According to the XML standard, users are allowed to define their own document type definition (DTD) files that define data contents and constraints. The production of document type definition file in fact is fixed by a set of well-defined syntax and grammar. Following the DTD standard, we build production rules for the geospatial application description and spatial data description respectively. Using the customized DTDs, we can produce data accordingly. Listing 3.6 is extracted from the DTD of geospatial application description that shows the composition of an Application element.

```
<!--This is main content of the application model language which
   describes an application as three components including
   a description, a pseudecode and a series of procedure -->
<!ELEMENT application (name, description, pseudocode, procedure*)>

<!ELEMENT name  (#PCDATA)>

<!ELEMENT description  (#PCDATA)>

<!ELEMENT pseudocode (list)+>

<!ELEMENT procedure (list)+>

<!ELEMENT list (#PCDATA)>
```

Listing 3.6 Extract from the DTD of geospatial application description shows the composition of the Application element.

This above Listing describes the composition of a geospatial application, which primarily includes four elements such as name, description, pseudocode and procedure. The following is description of each element:

- Name, it is the application name that we would like to document,
- Description, a short description of the application,
- Pseudocode, it is an element that provide the analysis approach for later implementation using different GIS. It is independent to any GIS. Within

one pseudocode entry, there are one or many sentences to state the analysis method, and each sentence would be put into one list element. In turn, a list element is a series of character data, i.e. text.

- Procedure, it is an element that dedicates to specify GIS, in which it is described by a series of list that in turn it is a series of character data.

Follow the data definition described in the previous Listing, we can develop the following XML file for application description. As shown below is Listing 3.7, which is an extract of application description, for each element defined in the above DTD, there is a pair of markup text that enclosed data value inside.

```
          <application>
            <name>Underground gas tank leakage (First Approach)</name>
            <description>Finding affected parcels if the gasoline tanks in all gas stations leak at the same
time</description>
            <pseudocode>
              <list>1.Retrieve from the database all gas stations.</list>
              <list>2.Generate buffer zones around them to simulate the area affected by a leakage.</list>
              <list>3.Overlay the zones and the parcels to find the affected parcels.</list>
            </pseudocode>
            <procedure>
              <list>GIS is MGE</list>
              <list>ulfbldr -v -I fton.dgn -E parcel.ulf -l 10 11</list>
              <list>tpdbr –v –A parcel.ulf  -O parcel.top </list>
               <list>qybdr –v –T parcel.top –I query1 –q "elements from parcel where PROP_TYPE=203;"
  </list>
               <list>tpzonr –v –T parcel.top –z "200 m" –O zone_A.top –I query1 –a</list>
               <list>tpdbr –v –T parcel.top –T zone_A.top –O result.top</list>
               <list>qybdr –v –T result.top –I query2 –q "elements from parcel overlaps elements from
  zone_A</list>
                <list>dgnbdr –v  -T result.top –D result.dgn –I query2</list>
             </procedure>
          </application>
```

Listing 3.7 Extracted from the XML of geospatial application description shows the information content of Application element.

As shown in the above Listing, the description shows all the four components. There are two groups of list element that show pseudo code and procedural operation respectively. In additional, the first list element within the procedure will record which GIS software can apply and the remaining list items are ready to run command for use with the software.

3.4 Detailed Design

The detailed design involves reconsideration of the object models and the functional model. The purpose of these operations is to simplify implementation and improve execution. In this section, we would enhance the object model by assigning identifiers, defining domain values. In addition, for simplifying and improving the later implementation, we would modify the object model accordingly.

3.4.1 Project Information

For each application that we would like to perform and output result for reporting, it should attach to particular task under particular project. In the object model of geospatial application, definitely, it can provide sufficient information for users to understand what and how the application can successfully be performed. Additional project related information is important, especially for information trace back and reporting. Project information relates to a geospatial application should includes project title, organization and date. Project title is a name of the main project that the application is belonging to. Subtask is an identifier for the application within in main project title. Organization is an identifier to record related party who designs and develops the underlying application. Finally, the date is to provide time information for the application.

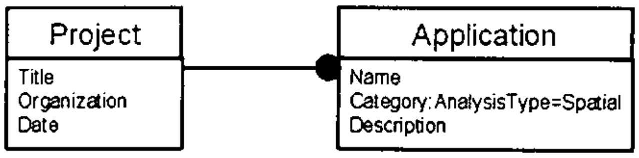


Figure 3.9 Object model of Application with the introduction of project information

Project information is part of the component of application information, and we could simply put all the designated attributes into the object. However, we would like to record the project information separately and introduce a new project object. The reason to this is because a project may associate to multiple application information. In the Figure 3.9, it shows an extraction of the entire object model with the introduction of project information object.

3.4.2 Object Identity

In order to implement the object model, it requires to introduce object identity for maintaining associations between objects. In our case, there are two levels of object identity. The first level of object identity maintains association between the two types of input file in the system, of which the first type is used for describing geospatial application and the second type, is used for describing geospatial data. The second level of object identity is used to maintain association between objects within a single file e.g. application information. To introduce this level of object identity, we would not assign explicit identifier for each object in the file. Instead, object identity will be assigned to each data element when each element markup and its data value are parsed into the system.

```
<!ELEMENT application_model (application+)>
<!ATTLIST application
            ID ID #REQUIRED
              CATEGORY ( SPATIAL | GRID | NETWORK | TERRAIN | IMAGE |MIXED ) "SPATIAL">
              ...
```

Listing 3.8 Declaration of attribute list for the application element

Following this strategy, at the file level, an identifier would be assigned to each application object and spatial data object. According to the specification of XML, there is an attribution list declaration, which allows an assignment of an attribute type called 'ID'. This attribute type as defined is a unique identifier dedicated for an element. No two elements can share a single ID within a document. Therefore, in the DTDs, we introduce an attribute list with ID for each element of application and spatial data information. Listing 3.8 shows the declaration of attribute list for the application element and Listing 3.9 shows the declaration of attribute list for the spatial data element.

```
<!ELEMENT spatialdata (idinfo)+>
<!ELEMENT idinfo (name, descript, bounding?, keywords, from_application?)>
<!ATTLIST idinfo ID ID #REQUIRED>
```

Listing 3.9 Declaration of attribute list for the geospatial data element.

The modifier after the attribute name provides information to the XML processor how it should react to the element. As shown in Listing 3.9, the

'#REQUIRED' means attribute 'ID' is an ID type attribute and is a required attribute for the application element. If the attribute is optional, the modifier would be '#IMPLIED'. Following the declaration of Listing 3.9, the following Listing 3.10 is an example of the XML data.

```
<application_model>
 <application ID="UGTL1" CATEGORY="SPATIAL">
....
</application>
```

Listing 3.10 Example shows the attribute data for application element.

On the other hand, to build associations after data is parsed into the system; the XML parser can handle this automatically without extra programming effort. As mentioned in Chapter 2, the XML parser can build a document model in memory and maintain element association as a parent and children relationship. By knowing the element name, one can retrieve its descendants or children elements. Each element may have different children element with different name, by comparing and further traversing with a known element name, element's attributes can be retrieved easily.

3.5 Summary

In this chapter, we showed the analysis and design of the application model. Based on the application model, we designed a geospatial application information browser, which enabled users to view the application model and allows connection to any geospatial data clearinghouse for searching geospatial metadata document. With the browser, users do not require to understand GIS well, but the system can help the users for solving any available spatial analysis and modeling problem.

CHAPTER 4
IMPLEMENTATION

4.1 Introduction

In this chapter, we will describe the use of file system for implementing the object model of the application model. In addition, based on the model, we will build the geospatial application information browser. The reason to use file-based implementation is because it is simple to implement with no extra cost, easy to understand and hence it can provides a concrete base for an experimental trial to build application without spend extra time on exploring other technology. The focus of this chapter is to show the implementation of application model and its proof of concept. In fact, we also considered other data management approach for the implementation such as using relational databases and object-oriented databases. Relational implementation could provide a comprehensive data management capabilities for it's underlying application. This approach requires to dissemble the developed object model into a series of tables and relational joins. Finally, this could fit the object model into the relational database, however modification is unavoidable and part of the semantics carried by the original object model might lose or duplication of information may happen. For the object-oriented database, object model is well fit into it. Nevertheless, with the additional data management facilities in transaction management, data recovery, object querying, concurrency control, security and so on, which makes it become complex and possibly diverse our focus to these areas. In fact, it is also not affordable for us to spend time for exploring the implementation technique using object-oriented database. On the other hand, file-based implementation provides definable production rules, which could provide structural information about the stored data such as hierarchical structure. With limited coding effort, it could easily rebuild the object model within the program life cycle and keep its basic structure persistent using the file system.

To implement the object model using file-based approach, it requires several steps to map the model to a file structure. Firstly, the object model should be organized into files, which need to decide particular classes and associations group into file. Secondly, file approach should be defined, in which whether ASCII file or binary file will be used. In our case, we would use ASCII file type. Thirdly, the approach to implement object identity should be defined to rebuild object model. Finally, we also need to define implementation of domain; attribute and associations.

We choose file-based implementation approach, which definitely meets these requirements and can be faster. In the following sections, we will describe how to implement the object model of application model using XML, how to build object model using DOM after the XML file is parsed into memory using XML parser. Then we will extract fragments of XML document and render it to form user interfaces and finally we will describe the interfaces to geospatial data clearinghouse for searching and retrieving metadata document.

4.2 Implementing the Application Model

We would use XML to implement the structure of the application model. As mentioned in previous section, it is required to consider what file type need to be developed, the file format, implementation of object identity and other information such as domain, attribute and association. To determine the file type required presenting the model, it could consider the characteristics of file access. In the application model, there are primarily two parts: application information and spatial data information. Between these two components, there are associations between them, in which the application part is the most significant, which helps users to understand the application. Meanwhile the spatial data part is a supporting part, which provide a keyword set for identifying spatial data relates to particular application without guesswork. In fact, a single record of spatial data could be used in multiple application information. To minimize duplication of data, we will create two file types for storing information, one for application information and another for spatial data information.

Relating to the file format, since the use of XML, we use ASCII format file. For implementing the object identity, there are two approaches as discussed in Chapter 3. The first approach is used to assign identity for objects within a single document. For example, within a keyword element, there are theme keyword elements and place keyword elements. In this case, we would not assign object identity to each element explicitly in the file. Instead, we use DOM for handling the assignment of object identity. After the file is parsed and validated with a document type definition, all the elements are treated as object with identity assigned.

The second approach assigns identity for each element, which requires linking to another document. In our case, we need to link up the application object and spatial data object together. To do this, we place an explicit ID in each spatial data record in the document for identification. In the application information document, for each application, there are entries for assigning an ID corresponding to particular spatial data record.

We are using file system to store all the information. If there are too many documents, it may not be practical to read and parse all the files into memory for processing. It is because all the data should reside in memory and this may degrade the performance of data processing if not enough memory is available. In our case, since there are limited data, it is feasible to use this explicit identification in the document for linking different documents.

In addition, domain, attributes and association within the application model could be described by XML syntax and implement as file structure. To implement the application model, a document type definition (DTD) should be developed that describe the structure of the application model. Recalling the application model developed in Chapter 3 and applying the XML syntax to build DTD that mentioned in Chapter 2, we can build the DTD as shown in Listing 4.1.

```
<?xml encoding="US-ASCII"?>

<!--Revision: 0.1 application.dtd
   This is a DTD for describing geospatial processing procedure language.
   This document type definition (DTD) is developed Department of
   Land Surveying & Geo-Informatics, The Hong Kong Polytechnic University
   under research topic "Develop an Application Model for Geospatial Data
   Clearinghouse" All rights reserved 1998-1999 -->

<!--Declare the parameter entities-->
<!ENTITY % id.global "ID ID #REQUIRED">
```

```
<!ELEMENT application_model (application+)>

<!--This is main content of the application model language which
   describes an application as three components including
   a description, a pseudecode and a series of procedure -->

<!ELEMENT application (project?, name, description, pseudocode, procedure*)>
<!ATTLIST application
          %id.global;
          CATEGORY ( SPATIAL | GRID | NETWORK | TERRAIN | IMAGE |MIXED )
"SPATIAL">

<!--Project element helps to provide and organize project information
     in a large scale GIS project which may involve multiple type of
     GIS analysis procedures.-->
<!ELEMENT project (title, subtask_name, organization, date?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT subtask_name (#PCDATA)>
<!ELEMENT organization (#PCDATA)>
<!ELEMENT date (day?, month, year)>
<!ELEMENT day (#PCDATA)>
<!ELEMENT month (#PCDATA)>
<!ELEMENT year (#PCDATA)>

<!ELEMENT description  (#PCDATA)>

<!ELEMENT pseudocode (list)+>

<!ELEMENT list (#PCDATA)>

<!--Either shell command script or gui operation can be entered.
   Hence, it does not permit a procedure to enter both method at a time.-->
<!ELEMENT procedure  (gis,gis_version,(shell | gui),result?)>
<!ATTLIST procedure
          NAME CDATA #IMPLIED>
<!ELEMENT gis (#PCDATA)>
<!ELEMENT gis_version (#PCDATA)>
<!ELEMENT gui (step)+>
<!ELEMENT step (menu_selection, panel_entry)+>
<!ATTLIST step
          %id.global;>
<!ELEMENT menu_selection (#PCDATA)>
<!ELEMENT panel_entry (title,(button_action|variable)*)>
<!ELEMENT button_action (#PCDATA | variable)*>
<!ATTLIST button_action
          TYPE (single_click | double_click | select) "single_click">
<!ELEMENT shell (statement)+>
<!ATTLIST shell
          TYPE CDATA #REQUIRED>
<!ELEMENT statement (#PCDATA | variable)*>
<!ELEMENT result (variable)+>
<!ELEMENT variable (name,value)>
<!ATTLIST variable
          LINK CDATA #IMPLIED
          TYPE (spatial | general) "spatial"
          DESC CDATA #IMPLIED
          FILENAME CDATA #IMPLIED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

Listing 4.1 Complete Document Type Definition (DTD) of the Application Model

In the Listing, several features from XML are used for defining entity declaration, attribute type, attribute domain and association within the application model. Entity declaration, as mentioned in Chapter 2, is a symbol for reuse within the DTD. In the example, % id.global is a symbol with definition "ID ID #REQUIRED". To describe association between elements, it uses content model such as <!ELEMENT date (day?, month, year)>, in which the element date is composed of three elements in sequence including day, month and year. In addition, element "day" is optional as denoted by "?". To illustrate how to define the document structure, consider an element declaration shown in Listing 4.2 that is extracted from the Listing 4.1.

```
<!ELEMENT variable (name,value)>
<!ATTLIST variable
LINK CDATA #IMPLIED
TYPE (spatial | general) "spatial"
DESC CDATA #IMPLIED
FILENAME CDATA #IMPLIED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

Listing 4.2 An example shows the declaration of Variable element.

In Listing 4.2, it is a declaration of element "variable", in which (name, value) is the content model and there is an attribute list defined. The attribute list of the variable element containing four attribute types that including LINK, TYPE, DESC and FILENAME. Not all the attribute types are required. In fact, if the attribute type is marked by keyword "#REQUIRED" or no keyword following the attribute type, the attribute must be presented within the tags in the document. If the attribute type is marked by keyword "#IMPLIED", it means the attribute is optional. In the example, only attribute type "TYPE" is required and there is a domain that list out all the possible entry value. If there were no value specified, the default value would be "spatial" in this example.

Similarly, by recalling the object model of spatial data, a DTD for spatial data model could be developed as shown in Listing 4.3.

```
<?xml encoding="US-ASCII"?>
<!--Revision: 0 spatialdata.dtd, xml4j,_1_1_4-->
<!ELEMENT spatialdata (idinfo)+>
<!ELEMENT idinfo (name, descript, bounding?, keywords, from_application?)>
<!ATTLIST idinfo ID ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT descript (#PCDATA)>
<!ELEMENT bounding (westbc, eastbc, northbc, southbc)>
<!ELEMENT westbc (#PCDATA)>
<!ELEMENT eastbc (#PCDATA)>
<!ELEMENT northbc (#PCDATA)>
<!ELEMENT southbc (#PCDATA)>
<!ELEMENT keywords (theme, place?, stratum?, temporal?)>
<!ELEMENT theme (themekey+)>
<!ELEMENT place (placekey+)>
<!ELEMENT stratum (stratkey+)>
<!ELEMENT temporal (tempkey+)>
<!ELEMENT themekey (#PCDATA)>
<!ELEMENT placekey (#PCDATA)>
<!ELEMENT stratkey (#PCDATA)>
<!ELEMENT tempkey (#PCDATA)>
<!ELEMENT from_application (#PCDATA)>
```

```
<!ATTLIST from_application
        application_ID CDATA #REQUIRED>
```

Listing 4.3 Complete Document Type Definition (DTD) of the Spatial Data Model

As shown from the Listings, structure of the object model can easily be described by the DTD. To provide links between the application model and the spatial data, for each variable element, there is an attribute list containing an entry for spatial data link as shown in Listing 4.4.

```
<!ATTLIST variable
        LINK CDATA #IMPLIED
        TYPE (spatial | general) "spatial"
        DESC CDATA #IMPLIED
        FILENAME CDATA #IMPLIED>
```

Listing 4.4 Declaration of attribute list of the Variable element for the Application Information

By comparing the value of the LINK attribute type, it allows us to retrieve a particular spatial data record from the spatial data document. With the DTDs, we can further develop a testing data for application model and spatial data model respectively. The following Listing 4.5 shows a testing data for application model.

```
<?xml version="1.0"?>
<!-- Revision: 0 java/applicaton/application.xml, xml4j_1_1_4 -->
<!DOCTYPE application_model SYSTEM "application.dtd">
<application_model>
 <application ID="UGTL1" NAME="Underground gas tank leakage (First Approach)" >
  <description>Finding affected parcels if the gasoline tanks in all gas stations leak at the same
time</description>
```

```
  <procedure>
   <statement>InformationManager.Open <variable LINK="BASEMAP" TYPE="spatial" DESC="A data set contains land parcels, gas station">layer 1</variable></statement>
   <statement>DBManager.Query().QueryCondition(select <variable TYPE="general" DESC="Property type">PROP_TYPE</variable> = <variable TYPE="general" DESC="Property id">203</variable> from <variable LINK="BASEMAP" TYPE="spatial" DESC="A data set contains land parcels, gas station">layer 1</variable>)</statement>
   <statement>InformationManager.PixelResolutionSettings(<variable TYPE="number" DESC="Pixel resolution">5</variable>).UserDefinedResolution.Value(<variable LINK="BASEMAP" TYPE="spatial" DESC="A data set contains land parcels, gas station">layer 1</variable>)</statement>
   <statement>InformationManager.Utilities.Zone.CreateBuffer.Settings(CreateOn:<variable TYPE="general" DESC="Feature selection">Current Selection</variable>, Buffer Width:<variable TYPE="general" DESC="Buffer distance">200</variable>, OutputZoneName:<variable TYPE="spatial" DESC="Output zone">ZONE_A</variable>)</statement>
   <statement>InformationManager.MapQuery.ByZone(ChooseZone:<variable TYPE="spatial" DESC="Input zone">ZONE_A</variable>, ChooseFreatureType:<variable TYPE="general" DESC="Feature Type">Polygon</variable></statement>
   <statement>InformationManager.Save.Zone(CurrentZone:<variable LINK="UGTLD1" TYPE="spatial" DESC="Input zone">RESULT_ZONE</variable></statement>
  </procedure>
 </application>
</application_model>
```

Listing 4.5 Example shows an XML document for the Application Information

In addition, the following Listing 4.6 shows a testing data for spatial data model.

```
<?xml version="1.0"?>
<!-- Revision: 0 java/applicaton/spatialdata.xml, xml4j_1_1_4 -->
<!DOCTYPE spatialdata SYSTEM "spatialdata.dtd">
<spatialdata>
<idinfo ID="BASEMAP">
        <name>Base map</name>
        <descript>A data set contains parcel and its attributes</descript>
        <keywords>
                <theme>
                        <themekey>underground gas</themekey>
                        <themekey>building</themekey>
                        <themekey>population</themekey>
                        <themekey>parcel</themekey>
                        <themekey>road</themekey>
                        <themekey>planning</themekey>
                </theme>
                <place>
                        <placekey>Canada</placekey>
                        <placekey>New Brunswick</placekey>
                </place>
        </keywords>
        <from_application application_ID="Digi">Manual Digitizing from hardcopy
maps</from_application>
</idinfo>
</spatialdata>
```

Listing 4.6 Example shows an XML document for the Spatial Data Information

The above Listing are XML documents for application information and spatial data information respectively. The common characteristic of the documents is that each element inside is composed of start tag, data and end tag. In addition, for each document, there is a single root element, which serve as an entry point to the document content. In our example, application information has a document root called application_model. Inside it, there could have multiple instances of application element. Similarly, in the document of the spatial data information, there is a document root called spatialdata. However, the XML documents and DTDs are only the implementation of the object structure. To implement the functional model, a XML processor is required.

4.2.1 The XML Processor

In order to process the information correctly, the information should be validated against defined document type definition (DTD). The technology that we use in this project to validate the information is a XML processor developed using JAVA by IBM, which is called XML4JAVA. This XML processor provides two main functions [IBM, 1998]:

- Parsing an XML document and construction of a Java object tree, and
- Generation of an XML document from a Java object tree.

In fact, the XML4JAVA is not only a XML processor with its operational behaviors as specified in the XML specification, but also an implementation of DOM. As mentioned in Chapter 2, DOM is a software specification used to build API for constructing an internal tree by reading a XML document. It allows the application software to navigate that tree. In fact, the DOM provides a series of interfaces for implementation, in which an interface is a collection of signatures to build methods with no information given about their implementation. XML4JAVA is an implementation of DOM. For instance, TXDocument from XML4JAVA implements Document from DOM.

The XML4JAVA [IBM, 1998], contains several packages, in which each package provides common groups of objects for processing XML documents. The following are packages available from the software:

1. com.ibm.xml.omake that provides class and exception for handling expression matching, hence it enables the processor to recognize special characters for constructing XML document such as ., *, +, ?, [, (,), |, \.
2. com.ibm.xml.parser that provides classes, interfaces and exceptions for parsing and handling a XML document. In addition, it provides implementation of the DOM interfaces.
3. com.ibm.xml.parser.util that provides classes to create and manipulate tree view of a XML document using Swing's JTree class. Swing is part of JAVA

Foundation Classes for building graphic user interface. In addition, it provides HTMLPrintVisitor class, which could print DOM and XML4JAVA defined nodes in HTML-like format.

4. com.ibm.xml.xpointer that provides classes, interfaces and exceptions for parsing XPointer expression, generating an XPointer instance from a node in a document tree, searching for nodes pointed by an XPointer instance. XPointer is a linking facility, which is used in XML document.
5. org.w3c.dom that provides interfaces and exceptions for implementing the DOM Level 1 classes. This package is developed by W3C and the com.ibm.xml.parser package contains classes and exceptions implement this.
6. org.xml.sax that provides classes and interfaces for building XML processor. It is an event-based implementation of XML processor, which is different from the tree-based implementation that DOM does. Using this approach, it is unnecessary to form a tree structure for a XML document. Instead, it would parse the XML document with different defined events notify to the application that implement it. For example, if the parser starts to read a XML document, it would trigger a start document event. Similarly, if the document ends, it would trigger an end document event. For particular event, there might be some handlers to be notified and handle the parse information accordingly. For example, if the parser starts to read an element, it triggers a start element event. There is an element handler is notified by this event and starts to analyze if the read element is equal to particular element type. If equals, print it on screen. Otherwise, discard it. This type of implementation is similar to

the event handling in a graphic user interface and has advantages in safe memory and provides efficient parsing if the size of XML document is very large. In addition, the XML4JAVA provides implementation of SAX in com.ibm.xml.parser.SAXDriver.

7. org.xml.sax.helpers that provides classes for simpler implementation in JAVA such as class to make a persistent copy of an attribute list and so on. Since it is not included in core distribution of SAX, it may not be available in other languages.

In order to use XML4JAVA to parse and validate a XML document, we need to create a com.ibm.xml.parser.Parser object with program code as shown in Listing 4.7.

```
InputStream is = new FileInputStream(xml_filename);
TXDocument doc = new Parser(xml_filename).readStream(is);
```

Listing 4.7 Program code for building a XML parser instance

The TXDocument class implements the DOM document interface and the doc created is a root of the document tree using DOM.

Once the document root is created, it provides an entry point, which enables traversing the document and accessing related methods to manipulate document objects. Basically, a document object model is composed of nodes and most of the common methods to do object manipulation, which are implemented from the Node interface of DOM. Common methods of Node interface are getFirstChild(), which returns a Node object that is the first child of the Node; getChildNodes(), which returns a NodeList that contains all the children Nodes of this Node; getAttribute(java.lang.String attrName), which returns a attribute value of a requested attribute type and so on. In fact, there are methods defined in DOM could be used in the functional implementation of the application model, which enables navigating and retrieving objects in the document tree. Besides, the document tree should contain information follows the DTD for application information.

4.2.2 Implementing the functional model

As mentioned in Chapter 3, the application model is composed of structure, attributes and operations for describing geospatial application. In previous sections, we have discussed the use of DTD to build structure, relationship between elements and assignment of attributes according to the object model of application model and spatial data model. In this section, we would make use of the DOM and XML4JAVA to implement operations for the model.

The functional model we discussed in Chapter 3 contains two categories of operations. The first category is typical data manipulation operations such as add, delete and update objects. The second category is data navigation and retrieval functions such as find object, retrieve object and so on. Since, in our application program that is a geospatial application information browser is used for data rendering and searching uses. All data manipulation operations would be handled by text editor manually, although it is possible to build a database application with the application model.

Besides, to enable the browser renders application information and performs searching for geospatial metadata properly, the second category functions are necessary to implement. Except the searching geospatial metadata function, all the other operations are defined in DOM. We can simply reuse corresponding operations from the DOM to retrieve particular object from the documents. For example, the following Listing 4.8 shows a generic operation that could retrieve a Node object with specifies node name.

```
Vector getDocumentNodes(Node node, String nodeName){

  if (node instanceof TXElement)
  {
    TXElement el = (TXElement)node;
    if (el.getName().equals(nodeName)){
      nodes.addElement(el);
      if (DEBUG)
        System.out.println("In SearchDoc: Get element name: " + el.getName());
    }
  }

  if (node.hasChildNodes()){
   NodeList nl = node.getChildNodes();
   int size = nl.getLength();
   for (int i = 0; i<size; i++) {
     getDocumentNodes(nl.item(i), nodeName);
   }
  }
  return nodes;
}
```

Listing 4.8 Generic program code to retrieve Node object based on specify node name

The above operation requires two parameters. The first parameter is the document that we would like to retrieve. The second parameter is the node name of the expected node. This operation will return a Node object if the node matches the provided node name. After the node is retrieved, further object traversing is possible without starting from the root node.

```
public String getID(){
/*
Find the Node if its name equal to "application".
Then return the attribute value if the attribute type equal to "ID"
*/
      if (AppNode.getName().equals("application"))
      return AppNode.getAttribute("ID");
      else return null;
  }
```

Listing 4.9 Operation to retrieve Node object based on specify node name

On the other hand, some cases may require retrieving all the attributes of a Node object. To retrieve an attribute from a Node, the first step is to retrieve the Node object. The second step is to retrieve the attribute value using a known attribute type name. The Listing 4.9 shows an operation, which could retrieve attribute value of type ID from an application node.

Depending on what information is retrieved from the document, operations shown in above Listing 4.8 and 4.9 could be reused to meet the purpose with only minor adjustment. This could be applied to find elements such as procedure, pseudo code, application and its attributes. In fact, the DOM and XML4JAVA provide most of the functional requirements of the application model.

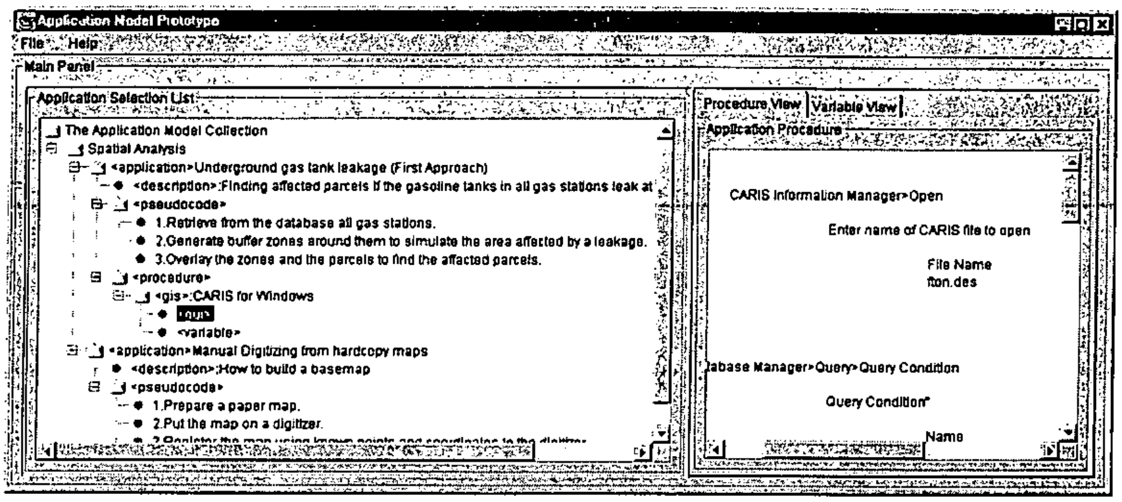
4.3 Implementing the user interface

In this project, we use Swing to implement the graphic user interface (GUI). Swing is a group of JAVA APIs providing GUI components such as combo box, tree, list box, text area and so on. In addition, it is a part of JAVA Foundation Classes (JFC). The building of GUI in fact is also an implementation of the dynamic model, which describe the interaction between the program and users over time. The basic design criteria of the GUI in this project is to enable users for traversing the application information easily and provide organized presentation of application information using different rendering techniques.

4.3.1 The main panel

After the XML documents for application information and spatial data information are parsed, corresponding document tree could be formed internally within the program life cycle. Then the main panel as shown in Figure 4.1 would be opened.

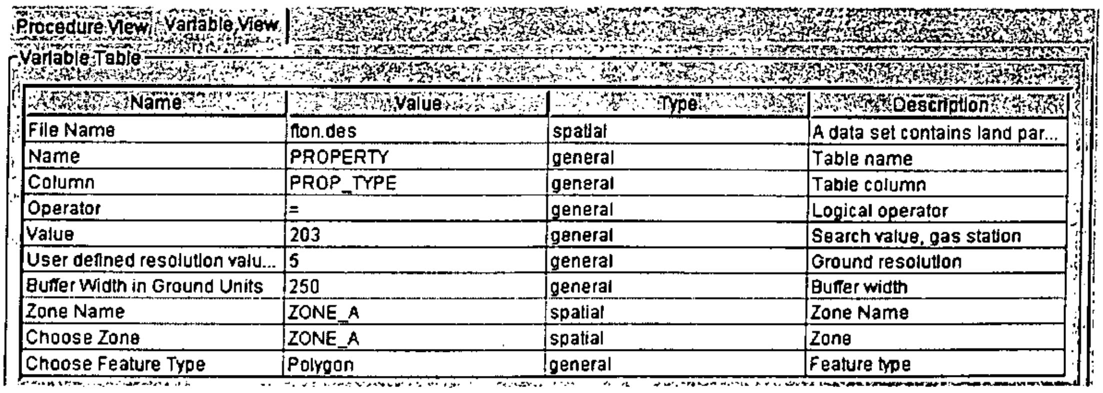
The main panel is composed of two sub-panels. On the left-hand side is a tree view. There are two node types in the view: non-leaf node and leaf node. The non-leaf node is represented as a open or closed folder and is expandable to view the underlying contents, in which it may contains other non-leaf nodes and leaf nodes. Meanwhile, the leaf node is an atomic data that is not expandable. In the tree view, all the application information is stored under a topmost folder called "The Application Model Collection". This folder could be opened by a mouse click and could be expanded to several folders that contains application information classified by its analysis type such as spatial analysis, grid analysis, image analysis, network analysis and mixed.



```
The Application Model Collection
  Spatial Analysis
    <application>Underground gas tank leakage (First Approach)
      <description>:Finding affected parcels if the gasoline tanks in all gas stations leak at
      <pseudocode>
        1.Retrieve from the database all gas stations.
        2.Generate buffer zones around them to simulate the area affected by a leakage.
        3.Overlay the zones and the parcels to find the affected parcels.
      <procedure>
        <gis>:CARIS for Windows
          <gui>
          <variable>
    <application>Manual Digitizing from hardcopy maps
      <description>:How to build a basemap
      <pseudocode>
        1.Prepare a paper map.
        2.Put the map on a digitizer.
```

```
CARIS Information Manager>Open
        Enter name of CARIS file to open
                File Name
                flon.des
tabase Manager>Query>Query Condition
        Query Condition*
                Name
```

Figure 4.1 The main panel of the geospatial application information browser.



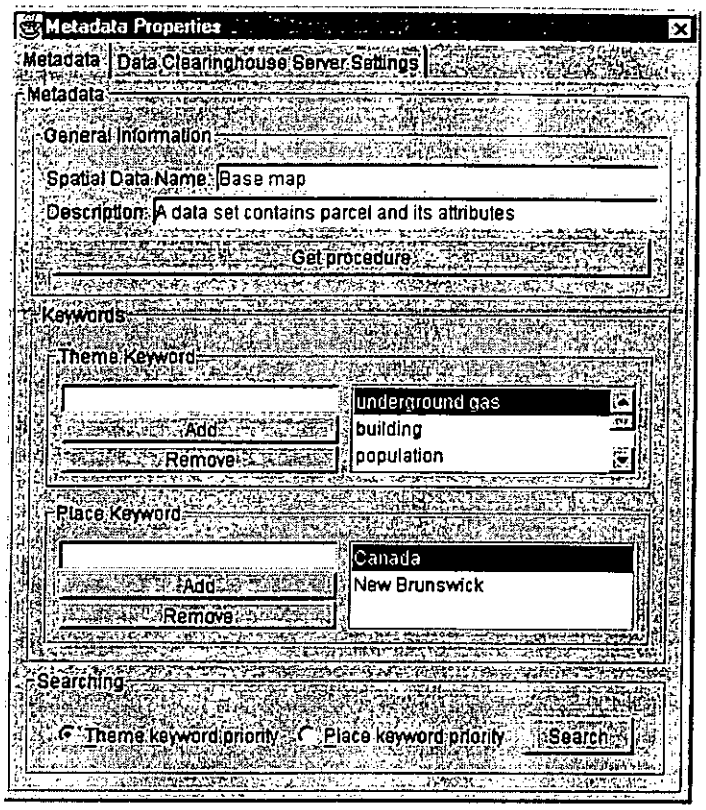
Name	Value	Type	Description
File Name	flon.des	spatial	A data set contains land par...
Name	PROPERTY	general	Table name
Column	PROP_TYPE	general	Table column
Operator	=	general	Logical operator
Value	203	general	Search value, gas station
User defined resolution valu...	5	general	Ground resolution
Buffer Width in Ground Units	250	general	Buffer width
Zone Name	ZONE_A	spatial	Zone Name
Choose Zone	ZONE_A	spatial	Zone
Choose Feature Type	Polygon	general	Feature type

Figure 4.2 Variable view of the geospatial application information browser.

If application of particular analysis type is available, users could open the folder to view the information. For each application, there are folders containing application description, pseudo code and procedure. Users could read the application description to determine if further browsing the information is necessary or not. While the pseudo code is a folder that could expand to a list of ordered statements. The statements tell the users how to performs the selected application. In addition, inside the procedure folder, users could find any available type of GIS that provide implementation of the pseudo code. Depending on the specific GIS, it would provide leaf-node called GUI or Shell, and users could retrieve the GUI procedure or commands to run the application and show it in the right-hand panel by a mouse click. Similarly, by clicking the variable leaf-node, a variable view rendered as table would show in the right-hand panel that lists out metadata of all involving variables such as its name, type, value and description. Figure 4.2 shows the variable view of the browser.

4.3.2 The Metadata Properties Panel

The variable view shown in Figure 4.2 could classify variables into two types: general and spatial. The general variables are those modifiers and its value that support to perform a command in command shell or an operation in GUI. The spatial variables instead are more complex that is a geographic data for processing. In the application information, it provides hint about is existence in the procedure. Consequently, to identify its metadata so that it enables the users to access it, we design another panel called Metadata Properties Panel. This panel could be accessed when a spatial type variable is highlighted from the variable view. Figure 4.3 shows the Metadata Properties Panel, which shows metadata information of a base map.



Spatial Data Name: Base map
Description: A data set contains parcel and its attributes

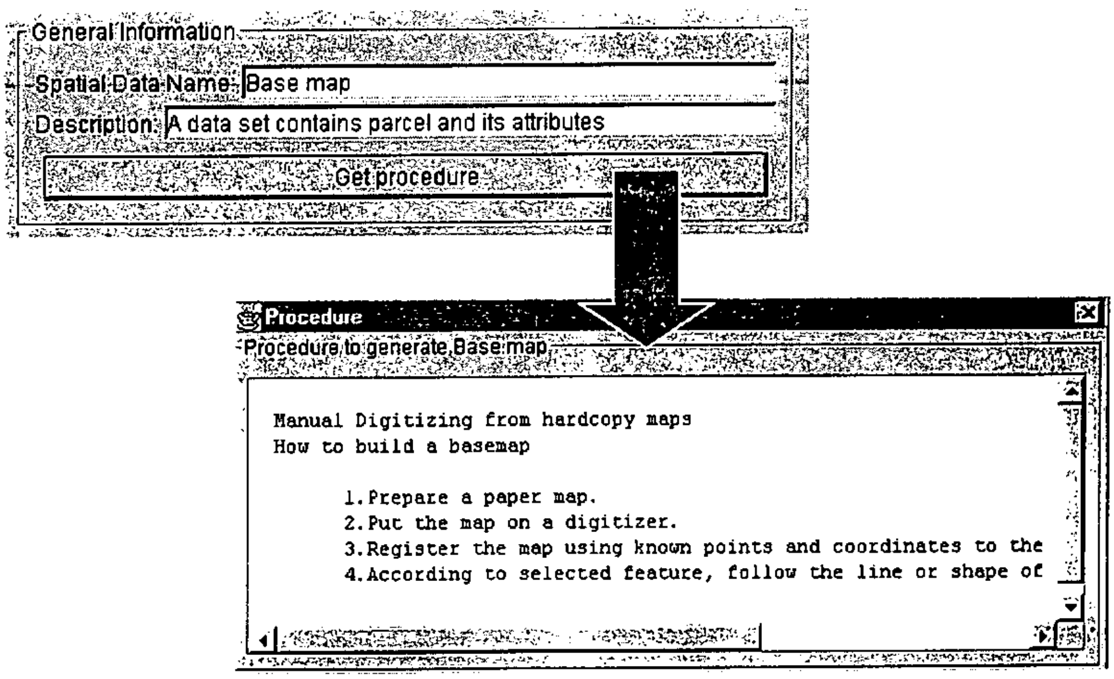
Theme Keyword: underground gas, building, population

Place Keyword: Canada, New Brunswick

(● Theme keyword priority) ○ Place keyword priority

Figure 4.3 Metadata properties panel of the geospatial application information browser

This panel will show limited metadata about the selected spatial variable in two parts. The first part with header "General Information", which provides a spatial data name and its description as well as a button that could show the procedure to generate the spatial data. In fact, for each spatial variable described in the application information, it contains a link that points to particular spatial data description placed in the spatial data information. The application information browser uses this link to retrieve the spatial data information and display it properly on the panel. On the contrary, for each spatial data, it contains link points to an application description that can generate it. Therefore, for each spatial data retrieved, it can show the application information that generates it if the application is available. Figure 4.4 shows the panel that displays the procedure to generate selected spatial variable.

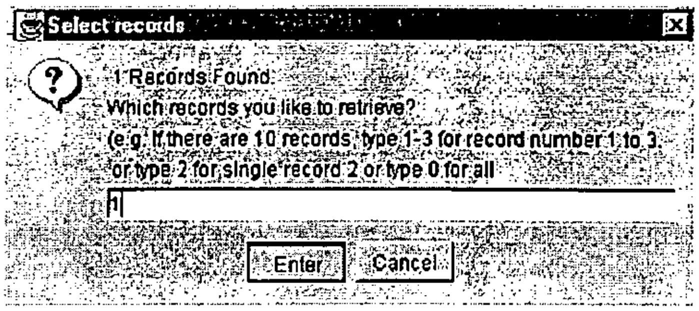


```
Manual Digitizing from hardcopy maps
How to build a basemap

        1.Prepare a paper map.
        2.Put the map on a digitizer.
        3.Register the map using known points and coordinates to the
        4.According to selected feature, follow the line or shape of
```

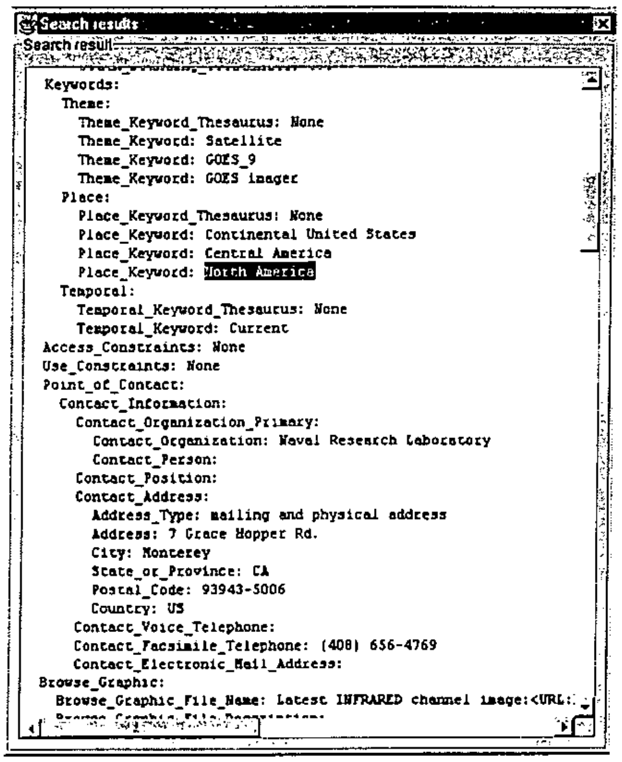
Figure 4.4 Procedures view for generating spatial data.

The second part of the metadata property panel provides two sets of keyword related to the selected spatial variable. It is expected that the keyword sets are entered by geospatial application designer and used for searching corresponding metadata document. This helps users to start searching the spatial data clearinghouse for metadata document using the provided keywords without guesswork.



```
1 Records Found.
Which records you like to retrieve?
(e.g. If there are 10 records, type 1-3 for record number 1 to 3
or type 2 for single record 2 or type 0 for all
1
```

Figure 4.5 The record selection dialog box



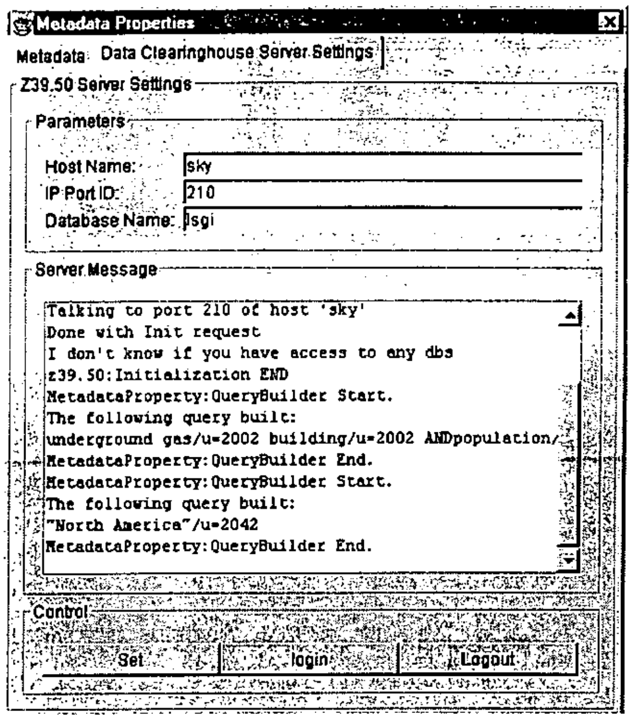
```
Keywords:
  Theme:
    Theme_Keyword_Thesaurus: None
    Theme_Keyword: Satellite
    Theme_Keyword: GOES_9
    Theme_Keyword: GOES imager
  Place:
    Place_Keyword_Thesaurus: None
    Place_Keyword: Continental United States
    Place_Keyword: Central America
    Place_Keyword: North America
  Temporal:
    Temporal_Keyword_Thesaurus: None
    Temporal_Keyword: Current
Access_Constraints: None
Use_Constraints: None
Point_of_Contact:
  Contact_Information:
    Contact_Organization_Primary:
      Contact_Organization: Naval Research Laboratory
      Contact_Person:
    Contact_Position:
    Contact_Address:
      Address_Type: mailing and physical address
      Address: 7 Grace Hopper Rd.
      City: Monterey
      State_or_Province: CA
      Postal_Code: 93943-5006
      Country: US
    Contact_Voice_Telephone:
    Contact_Facsimile_Telephone: (408) 656-4769
    Contact_Electronic_Mail_Address:
Browse_Graphic:
  Browse_Graphic_File_Name: Latest INFRARED channel image:<URL:
```

Figure 4.6 An example of retrieved metadata document displayed in a new window

However, if the provided keyword sets cannot satisfy users' requirement, there are add and remove button that allows users to enter and delete keyword to suit their needs. The type of keyword provided in the application include place keywords and theme keywords. At the bottom of this panel, there are options to modify the search criteria and a button available to start searching. If records found, a record selection dialog box will display to ask for user confirmation and then display the entire metadata document in a new window. Figure 4.5 and Figure 4.6 show the record selection dialog box and an example of retrieved metadata document respectively.

4.4 Implementing the interface to the geospatial data clearinghouse

To enable the capability for searching and retrieving actual geospatial metadata document, an interface to the geospatial data clearinghouse need to be to developed. It is expected that with the keyword sets provided, which point to place keyword and theme keyword section of CSDGM, it is promising to show the concept that help user to identify appropriate data set. Meanwhile, it is a fact that most of the geospatial metadata documents is available from the data clearinghouse network which uses ANSI Z39.50 protocol to communicate. Figure 4.7 shows the interface for setting up connection to geospatial data clearinghouse. To establish Z39.50 connection using the interface, it requires three parameters. The first parameter is host name of the Z39.50 server or the geospatial data clearinghouse server. The second parameter is the IP port number for the Z39.50 connection. Usually, it will set to 210, which is a formal port number for Z39.50 connection. The third parameter is the database name.



Host Name: sky
IP Port ID: 210
Database Name: jsgi

```
Talking to port 210 of host 'sky'
Done with Init request
I don't know if you have access to any dbs
z39.50:Initialization END
MetadataProperty:QueryBuilder Start.
The following query built:
underground gas/u=2002 building/u=2002 ANDpopulation/
MetadataProperty:QueryBuilder End.
MetadataProperty:QueryBuilder Start.
The following query built:
"North America"/u=2042
MetadataProperty:QueryBuilder End.
```

Figure 4.7 User Interface for establishing connection between the Geospatial Application Information Browser and geospatial data clearinghouse.

4.4.1 The Z39.50 Client Requirements

Requirements of Z39.50 could be found from the specification of ANSI/NISO Information Retrieval Service and Protocol. There are core and optional functionality for communicating server machine using the service and protocol. In our project, we implement the core functionality, which consists of five necessary functions to complete data searching and retrieval as shown in Table 4.1.

Core Functionality	Description
Connection	Establishing a Z39.50 connection.
Initialization	Negotiating with the Z39.50 server for what services is available from it.
Searching	Searching the database.
Present	Retrieving the result from database.
Disconnection	Close the Z39.50 connection.

Table 4.1 Core Functionality for building Z39.50 client

We use a Z39.50 client API developed by the Online Computer Library Center (OCLC) for building the Z39.50 part in our application. The API is available from public domain and provides basic function to meet the requirements of core functionality.

On the other hand, for communicating between a client and a server using the Z39.50 protocol, both the client and the server should understand how to create Z39.50 messages. A Z39.50 message consists of two logical parts: Definition of the contents, which is defined by Abstract Syntax Notation1 (ASN.1) and definition of the encoding rules, which is defined by Binary Encoding Rules (BER). [OCLC 1998] Hence, for each Z39.50 message, it is described using ASN.1 in the specification. To understand the message and make it human readable, it should be decoded accordingly. Since the details of ASN.1 and BER are out of the project scope, we have applied it in our application for building Z39.50 messages, instead of further exploring these topics.

4.4.1.1 Connection

The establishment of Z39.50 connection requires the use of sockets. A socket is software interface for providing functions to establish TCP/IP connection, send and receive messages in a network environment. The use of TCP/IP because it is widely used by Z39.50 servers in the Internet. In order to form TCP/IP connection, there are two parameters are required: IP address and port number. IP address or so-called Internet address is an identification of machine on the network. It is actually a 32-bit number and usually written in 4 decimal numbers, in which each number represent by 8 bits of the address e.g. 158.132.27.55. Port number is used to tracking different conservation between client and server. For Z39.50 connection, the port number is usually assigned as 210. However, it depends on the Z39.50 server settings, the port number could be changed e.g. 6888. In our application, users to make the Z39.50 connection should enter both the server's IP address and port number.

4.4.1.2 Initialization

After the TCP/IP connection is established, it could start to communicate in Z39.50 protocol. Before any searching and retrieving operation can be made, initialization is required prior. Initialization allows the client to ask for supported facility from the server, since there are optional functionality that may not support by particular machine. This operation is usually called InitRequest and there are 9 parameters that are available for formulating an initialization. The most significant parameters in this case are the option settings, preferredMessageSize and maxmumRecordSize. The option settings allow the client to negotiate with the server what functionality is available, in which there are 14 options. Meanwhile, the preferredMessageSize and maxmumRecordSize tells the server the size of transferable message in normal case and the size of maximum message in exceptional case. According to the specification, this mechanism allow the transfer of normal length records i.e. preferredMessageSize to proceed in a routine fashion with easier buffer control, while this also provide flexibility for transferring occasional exceptionally large record without requiring the server to allocate buffer space for worst case records. In this way, data retrieval involves multiple records should not exceed the preferredMessageSize, otherwise a diagnostics message would be sent to the client for notification. Instead, it allows retrieving single record each time and providing that the record should not exceed the maxmumRecordSize, otherwise it would be substituted by a diagnostics message. Both parameters should be specified in bytes and in our case, we adjusted the preferredMessageSize (i.e. 500kb) larger than the typical size of a geospatial data document i.e. 28kb.

After the initialization request is encoded and submitted to the server, the client could ask for respond by another request usually called InitResponse. To ask for response, the client would pass a variable to the server, then the server would return it to the client to tell whether the InitRequest operation succeeded and accepted. If the returned value is a non-zero value, then a Z39.50 session is successfully established and it allows for starting the search.

4.4.1.3 Searching

Similar to initializing a Z39.50 connection, to start a Z39.50 searching requires two steps: search request and search response. The search request is an operation, which submits a Z39.50 message to the server for starting the search process. The message contains 20 parameters about the characteristic of query and its result. Although not all the parameters are required and it should compose of result set name, query type, query statement, database names and preferred record syntax to be a valid search request message. In addition, there is a mechanism to define a richer semantic query in Z39.50. This allows client to create a query that each query subject could associate with multiple attributes. The query subject in fact is a keyword and it is called term in Z39.50 specification. This term could have additional information, which qualify it as a title or heading. The additional information or qualifier is called attribute. A group of related attributes under the same theme could be formed as an attribute set. For any one who wants to develop a new attribute set, there is a starter attribute set called bib-1 as basis and it is identified by an Object Identifier (OID), i.e. 1.2.840.10003.3.1. The bib-1 provides a list of attributes available for building query. In fact, for any attribute set, the OID should be started with 1.2.840.10003.3. Users could develop their own attribute set based on the bib-1. In conjunction with geospatial metadata, there is a GEO attribute set, which is one of the components in GEO Profile has OID 1.2.840.10003.4.

In addition, for each attribute set, it could contain various attribute types. In bib-1, there are six attribute types including Use, Relation, Position, Structure, Truncation and Completeness. In each of these types, there are various attributes available to qualify query term. For GEO attribute set, there are three attribute types available including Use, Structure and Relation. In fact, available Use attributes are short names as stated in the CSDGM. For example, idinfo as Identification Information, descript as Description and so on. The structure type contains attributes such as date, coordinates, year and so on, while relation type contains attributes such as less than, equal, greater than and so on. Besides, for each attribute type and attribute set, it is identified by an integer. For example, Use is attribute type 1 and Structure is attribute type 4, while idinfo is 3100 and descript is 3102 as specified in GEO attribute set.

4.4.1.3.1 Result Set Name

Result set name is a name that assigns to a result set. A result set would be created automatically after a query is submitted to the server. In fact, it would be used for retrieving records by subsequently referenced by retrieval facility. It depends on the server if it support multiple result set to be created from client. Nevertheless, Z39.50 should support a result set name "default", which is also used in our implementation.

4.4.1.3.2 Query Type

The query type tells the server which query syntax would be used to create query statement. There are totally six query types available from the specification and not all the server would support all the type. The Table 4.2 shows the query type defined in the Z39.50 specification.

Query Type	Description
Type-0	User-defined query type, which allows using organization's developed query grammar.
Type-1	Reverse Polish Notation (RPN) query type, which is a mandatory supported type in Z39.50.
Type-2	Defined by ISO Common Command Language (ISO 8777), which is unusual to be used in Z39.50.
Type-100	Defined by ANSI/NISO Common Command Language (Z39.58), which is closely related to Type-2 query type.
Type-101	Extension of Type-1 to support proximity searching.
Type-102	Still underdevelopment and will support ranking.

Table 4.2 Query Type defined in the Z39.50 Specification. (Z39.50 Specification)

In our project, we use type-1 query, since the Z39.50 server i.e. Isite, that we use support this query type. Type-1 is the Reverse Polish Notation (RPN) query, which is used to create boolean queries and has structure that operand is specified prior to operators. For example, to create a query statement for a union of sets A and B, the RPN query is "A B AND", instead of "A AND B", where "A", "B" are operands and "AND" is operator.

4.4.1.3.3 Query Statement

A query statement in Z39.50 is composed of query term, attributes and operators. The formation of a query statement is starting with a term, then following by a slash '/' and a Use attribute. Optionally, it can be followed by a Structure attribute and a Relation attribute. For example the following are valid query statements, GIS/21, that is GIS as title, vegetation/3102/2, that is vegetation as description and a structure of word. It also depends on the query type to place the operators in appropriate position before or after the operand. In this project, a query statement is formulated with provides theme keywords and place keywords. For multiple keywords, we use operator "OR" to combine it. For example, if there are two theme keywords such as geology, superficial geology. The query statement as follow the rule of RPN, it would be presented as geology/2002 "superficial geology"/2002 OR. The integer 2002 means theme keyword.

4.4.1.3.4 Database Names

Z39.50 server could support multiple databases and the client could identify the database sets by providing their names in the search request message. However, not all the servers provide multiple databases. To determine this, client should use Explain facility to retrieve about the server implementation such as available databases.

4.4.1.3.5 Preferred Record Syntax

Preferred record syntax controls the encoding method for responding records from the server and also presenting record on the client. According to the GEO profile, it supports three record syntaxes. In this project, we use a record syntax called Simple Unstructured Text Record Syntax (SUTRS) which could present the returned record correctly on client interface.

After a search request is submitted to the server, the client could use search response message to retrieves information from the server for the query status. If the response is a non-zero value, the search is succeeded. By analyzing the responded message, the client could retrieve information including number of records found, search status and so on. This helps the client to further determine if all the records should be retrieved from the database and presented on the client.

4.4.1.4 Present

Present is one of the two services available from the retrieval facility as mentioned in the Z39.50 specification. Another service is called segment, which enables segmentation of record to avoid data transmission abort due to large message size. In the implementation, it would not support segmentation, but the present service. Present involves two messages to be submitted to the server as usual: Present Request and Present Response.

The Present Request is a message that asks the server which records from result set to be retrieved. It involves 10 parameters, but it should consist of result set name, result set starting point, and number of retrieved records. Result set name is set as default so that it could points to the result set from the search response. Result set starting point is a relative number in a result set, which accompany with the number of retrieved records could allows client to retrieve a range of records.

The Present Response is a message that asks the server for the status of the Present Request and returned record is available. It involves 6 data objects, in which there are 3 data objects are mandatory to be returned to client, which including number of records returned, next result set position and present status. If expected records are available, the message would also contain the respond records. The number of records returned is the number of successful retrieved records according to the Present Request, while the next result set position is a relative number of the result set that corresponding to next record of the last record retrieved from the Present Response. If the last record retrieved is also the last record of the result set, the next result set position is zero. In normal case, to determine whether a Present Response success or not, it would check the Present Status in advanced. Present Status is a status descriptor, which contains 6 values to indicate the status of the Present Response. The values are including success, partial-1, partial-2, partial-3, partial-4 and failure. The following Table 4.3 summaries the available status in Present Status.

Status	Description
Success	All of expected records are retrieved.
Partial-1	Parts of the expected records are retrieved due to termination by access control.
Partial-2	Parts of the expected records are retrieved due to preferred message size exceeded.
Partial-3	Parts of the expected records are retrieved due to termination by resource control at the client.
Partial-4	Parts of the expected records are retrieved due to termination by resource control at the server.
Failure	No record returned. Diagnostic information would be returned instead.

Table 4.3 Available status for Present Status (Z39.50 Specification)

Hence, after the Present Status is being examined to be success, contents of the respond record, which is BER encoded, would be decoded and rendered on client interface for display.

4.4.1.5 Disconnection

To disconnect the Z39.50 connection, we simply close the socket and then the server would notify the disconnection from client. Record set resides in server would be automatically deleted and the server would restate to wait for client connection again.

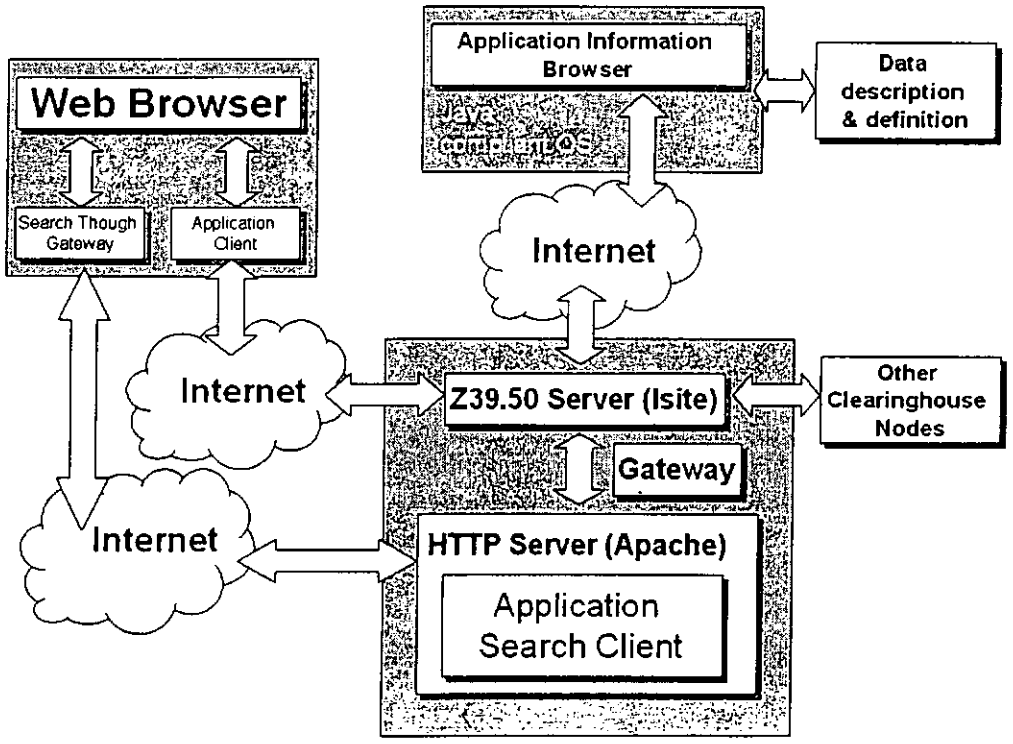


Figure 4.8 System architecture of the testing suite

4.5 Testing

We have setup a data clearinghouse system to test and demonstrate the concept and feasibility of the application browser. The clearinghouse server that we used is the Isite server described in Chapter 2, and it is a Z39.50 server. The server is running on an Intel-based Linux machine to connect to the PolyU campus network. In addition, we have setup an Apache web server with HTTP-Z39.50 gateway running on the same Linux machine to enable HTTP users for viewing and accessing the underlying database.

In fact, two kinds of client could access the server machine by using Java applet with Z39.50 implementation and ordinary web browser, i.e. Netscape or Internet Explorer. For our application, it can talk to the Isite server and retrieve information directly. Figure 4.8 shows the architecture of the testing suite in this project.

4.5.1 Application Scenario

We can use the geospatial application as described in section 3.2 to illustrate an application scenario of the Geospatial Application Browser. The geospatial problem to be solved is to find affected parcels if the gasoline tanks in all gas stations leak at the same time.

If you would like to solve the problem and do not know how to do the analysis, you can browse an application model that can tackle the problem using the geospatial application browser as shown in Figure 4.1. Once an application model with title "Underground gas tank leakage" is identified. You can expand the node by clicking it. Then, application information including description, pseudo code and procedure will be displayed. By reading the pseudo code, you can get the idea to tackle the geospatial problem. To find out if any software specify procedure available, you can expand the procedure node. On which, it would tell you the procedure of which GIS package is available. In our case, procedure that uses CARIS for Windows is available. Since CARIS provides graphical user interface. The following procedure will be displayed.

```
CARIS Information Manager>Open

                    Enter name of CARIS file to open
                              File Name
                              fton.des

CARIS Database Manager>Query>Query Condition

                    Query Condition
                              Name
                              PROPERTY

                              Column
                              PROP_TYPE

                              Operator
                              =

                              Value
                              203
Add
OK

CARIS Information Manager>Options>Resolution

                    Pixel Resolution Settings
                              User defined resolution value (Ground Units)
                              5
OK

CARIS Information Manager>Zone>Create>By Corridor

                    Zone Corridor Settings
                              Buffer Width in Ground Units
                              250

                              Zone Name
                              ZONE_A

CARIS Information Manager>MapQuery>By Zone

                    Map Query by Zone
                              Choose Zone
                              ZONE_A
```

```
                         Choose Feature Type
                         Polygon
OK
```

Listing 4.10 The application procedure for finding affected parcels if the gasoline tanks in all gas stations leak at the same time

If you expand the variable node, a variable list will be displayed as shown in Figure 4.2. For those spatial type variables, you can select it and the Metadata Properties Panel will be displayed, on which a set of keyword to identify the selected variable will be shown. Since the keywords provided might not fit to your requirement, you add or remove the keyword values. Once all the keywords are ready, you can connect to any geospatial data clearinghouse and use metadata search function to find an appropriate spatial data set.

4.5.1.1 Underground gas tank leakage

The following illustrate the step to implement the application procedure in Listing 4.10.

1. Open the CARIS Information Manager.

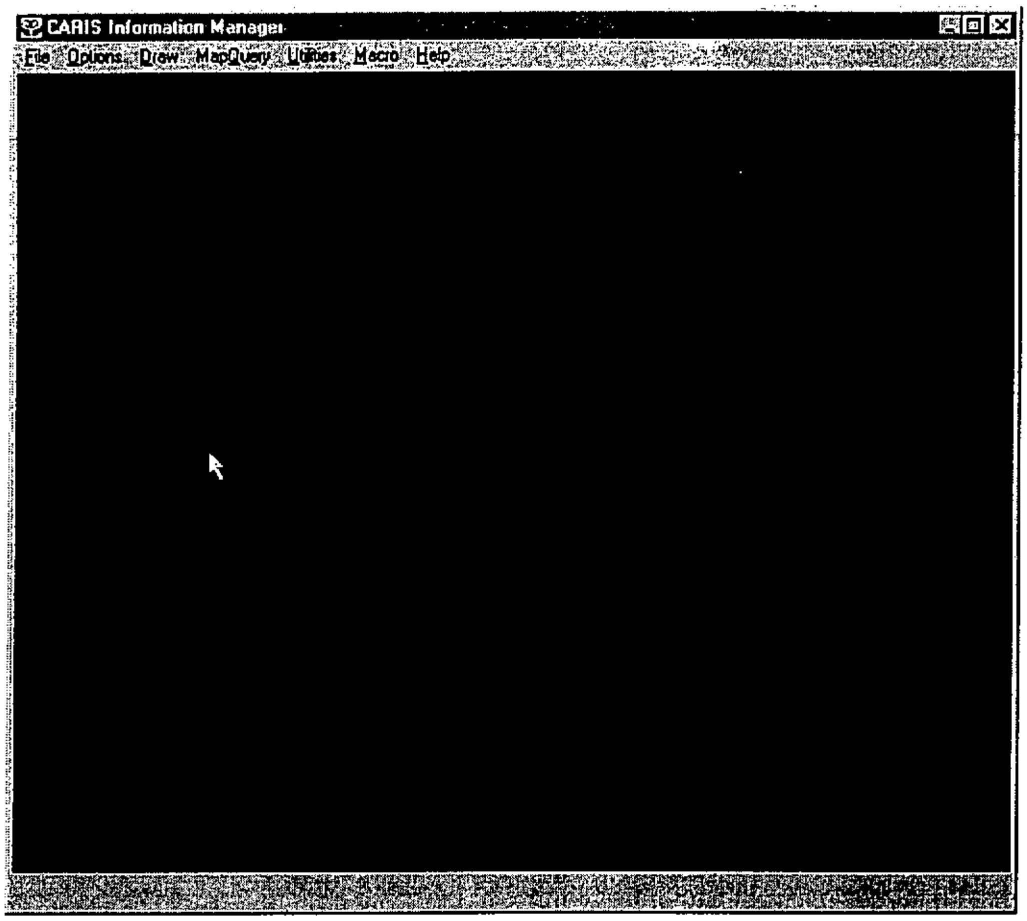
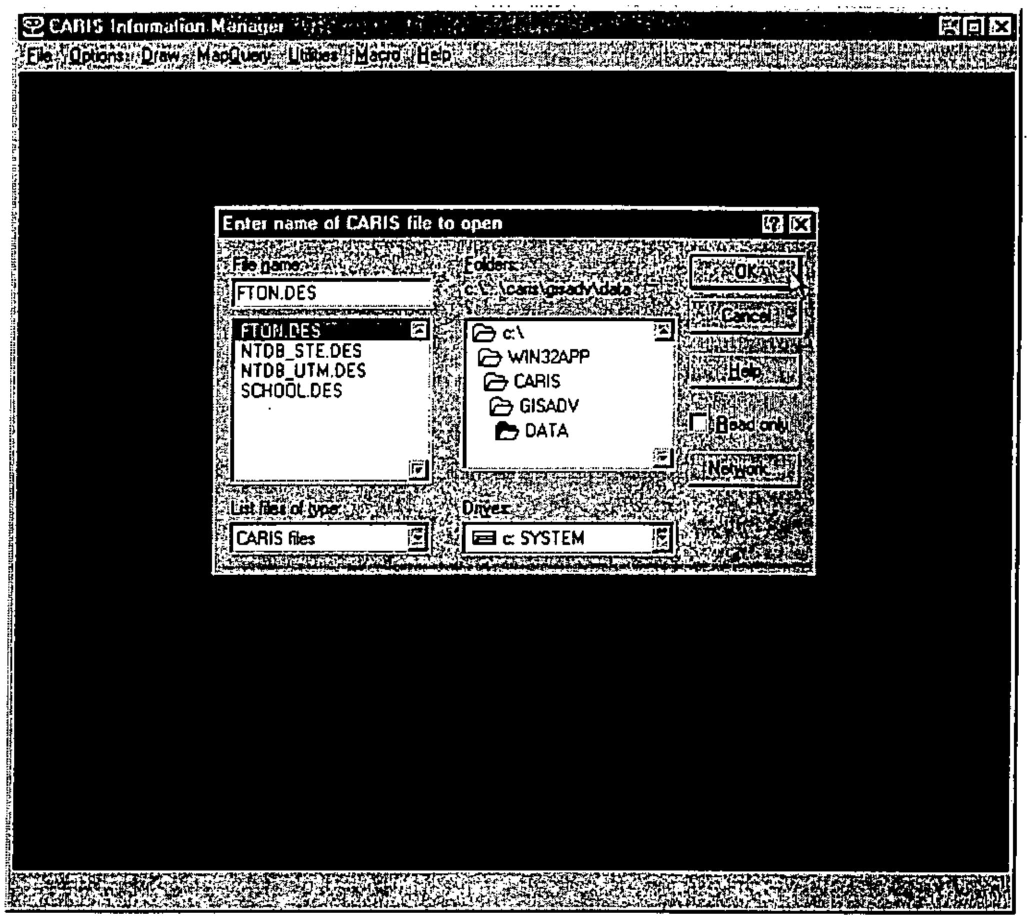

CARIS Information Manager
File Options Draw MapQuery Utilities Macro Help

Figure 4.9 The CARIS Information Manager

2. Click **File>Open**, a file open dialog will display. Then select fton.des and click **OK** to open the file.


CARIS Information Manager
File Options Draw MapQuery Utilities Macro Help
Enter name of CARIS file to open
File name:
Folders:
OK
Cancel
Help
Read only
Network...
List files of type:
Drives:

FTON.DES

c:\caris\gisadv\data

FTON.DES
NTDB_STE.DES
NTDB_UTM.DES
SCHOOL.DES

c:\
WIN32APP
CARIS
GISADV
DATA

CARIS files

c: SYSTEM

Figure 4.10 File open dialog opened.

3. Click **File>Open**, a file open dialog will display. Then select fton.des and click **OK** to open the file. The file will be opened and displayed on the map canvas as shown in Figure 4.11.

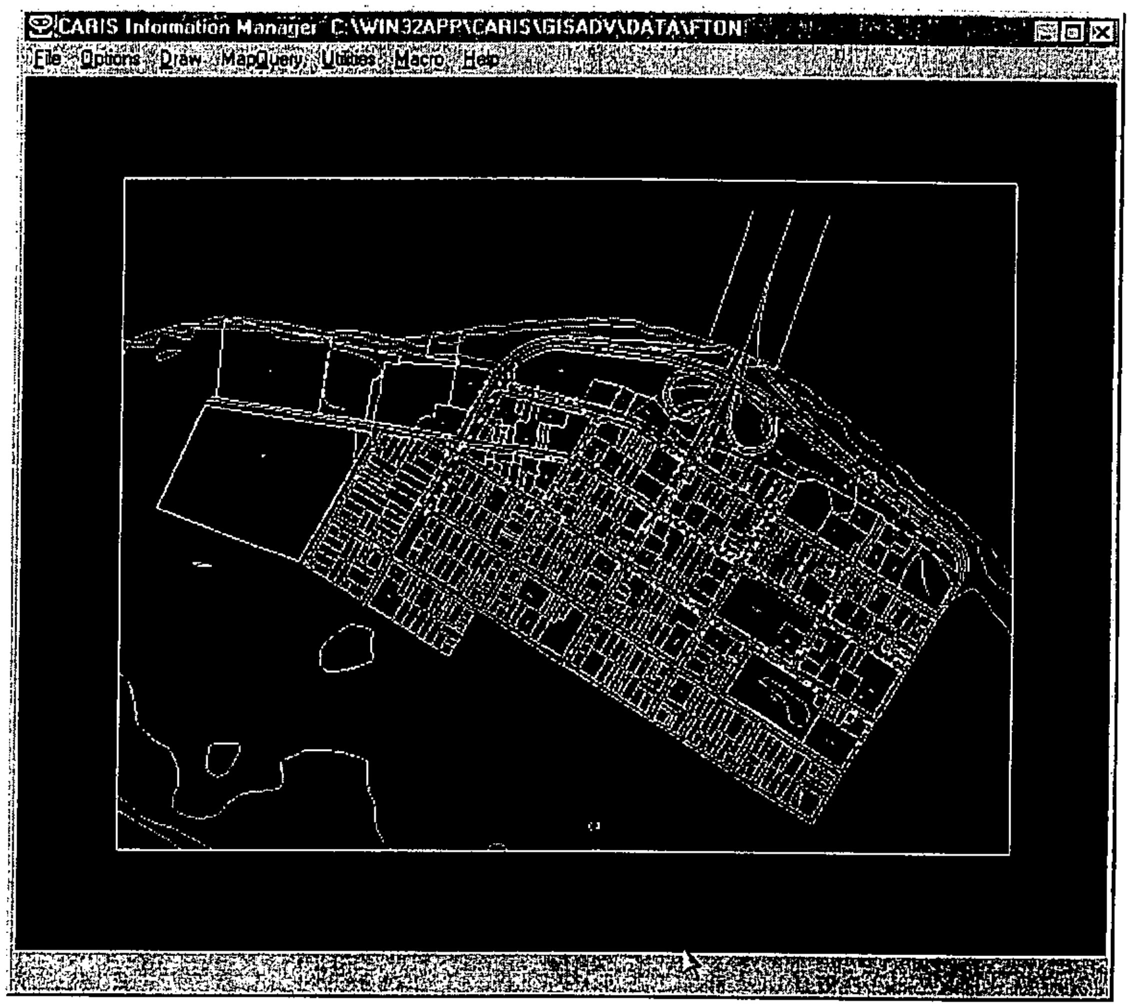

CARIS Information Manager C:\WIN32APP\CARIS\GISADV\DATA\FTON
File Options Draw MapQuery Utilities Macro Help

Figure 4.11 File open and display in the map canvas.

4. Open the CARIS DB Manager.

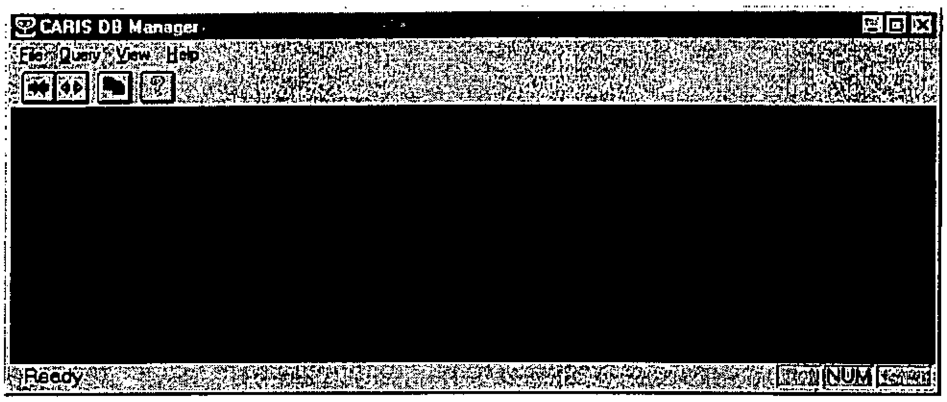
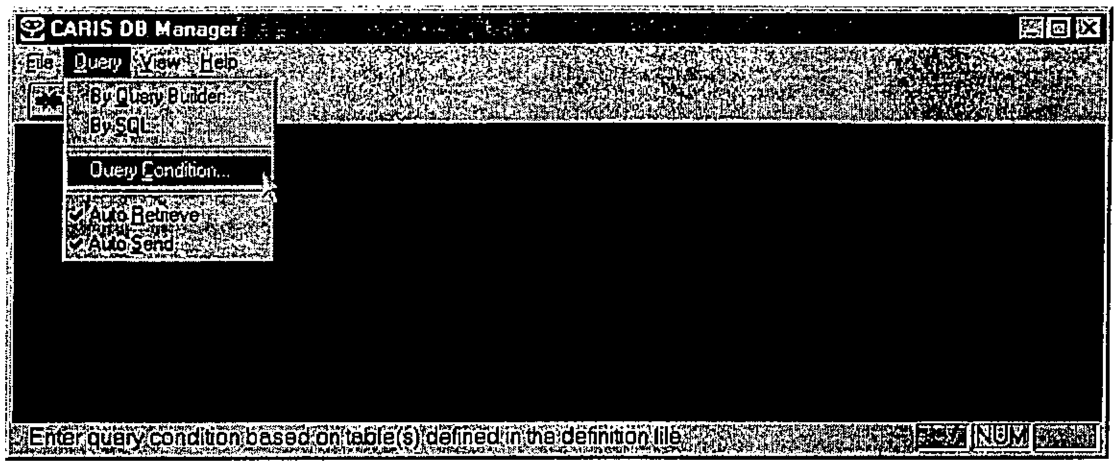


Figure 4.12 CARIS DB Manager.

5. Click **Query>Query Condition** to open Query Condition dialog box.



Enter query condition based on table(s) defined in the definition file

Figure 4.13 Open Query Menu.

6. In the Query Condition dialog box, enter column name, operator and value to construct a query for all underground gas station. Click **Add** and the query statement will be displayed.

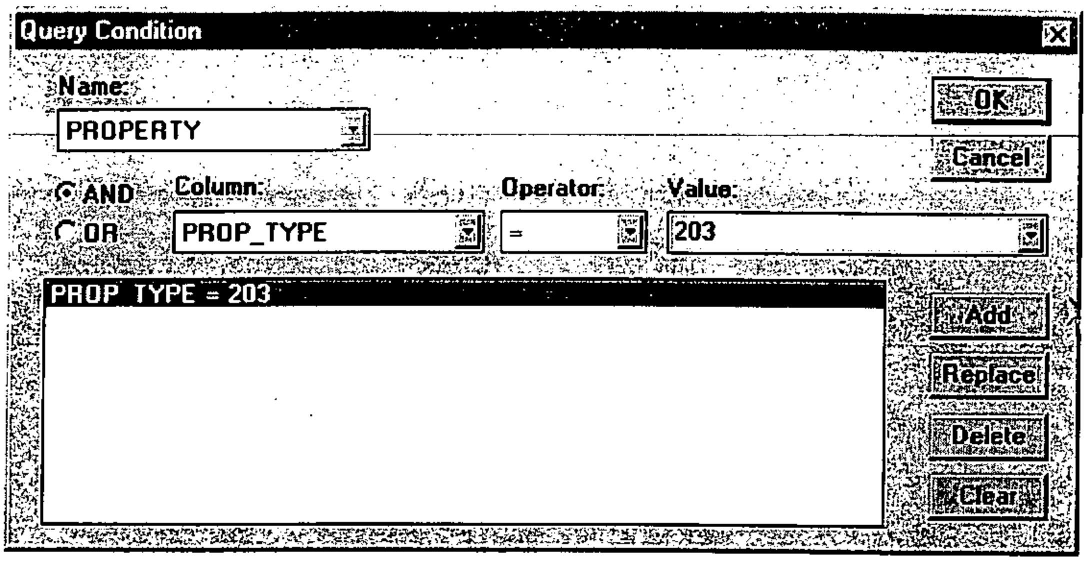
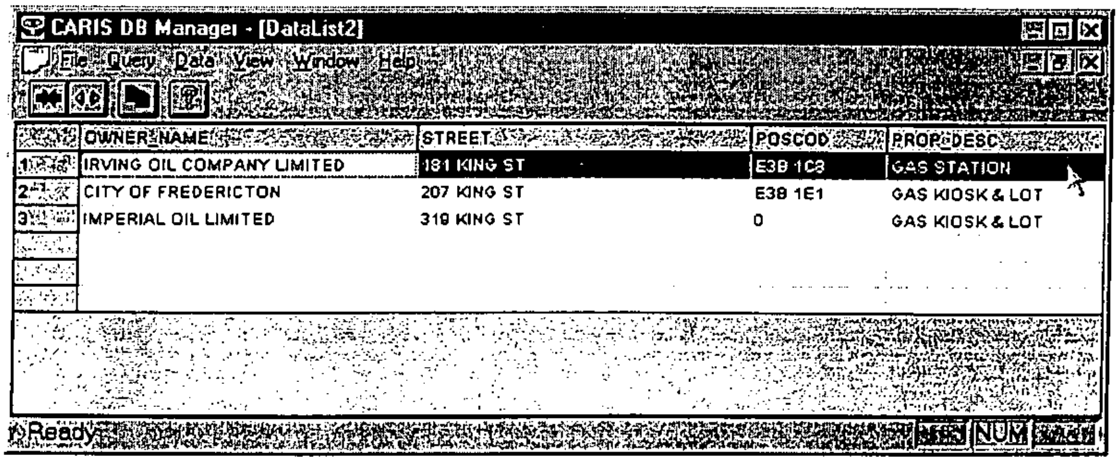


Figure 4.14 Use Query Condition Dialog to construct query statement.

7. Click **OK** to dismiss the Query Condition Dialog Box and query result will be displayed in the CARIS DB Manger as shown in Figure 4.15. There are three gas stations found.



	OWNER_NAME	STREET	POSCOD	PROP_DESC
1	IRVING OIL COMPANY LIMITED	181 KING ST	E3B 1C3	GAS STATION
2	CITY OF FREDERICTON	207 KING ST	E3B 1E1	GAS KIOSK & LOT
3	IMPERIAL OIL LIMITED	319 KING ST	0	GAS KIOSK & LOT

Figure 4.15 Query Results.

8. In the CARIS Information Manager, click Utilities>Zone>Pixel Resolution. A Pixel Resolution Dialog will be displayed.

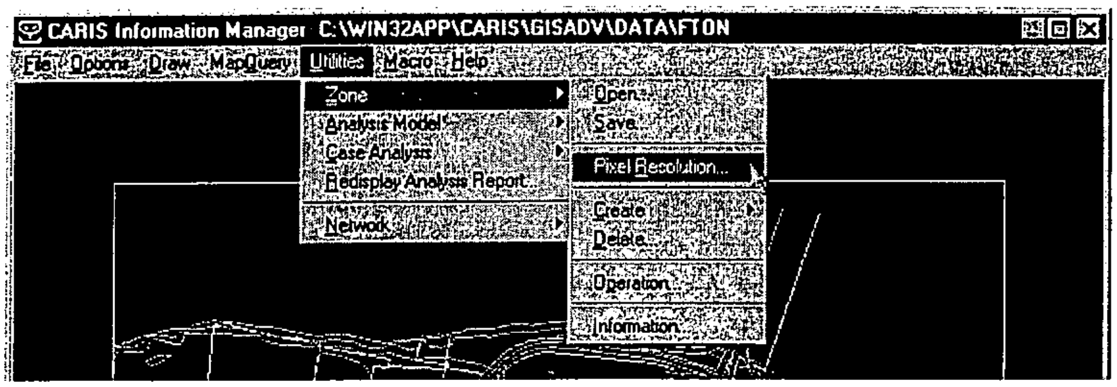
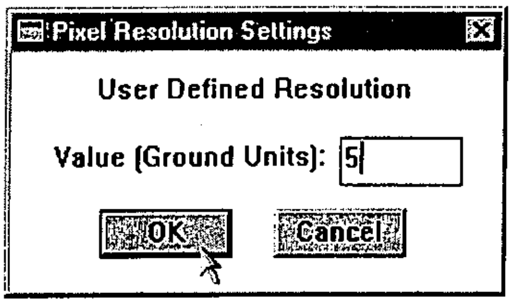


Figure 4.16 Open Pixel Resolution Settings Dialog Box.

9. In the Pixel Resolution Dialog box, enter 5 then click OK to dismiss it.

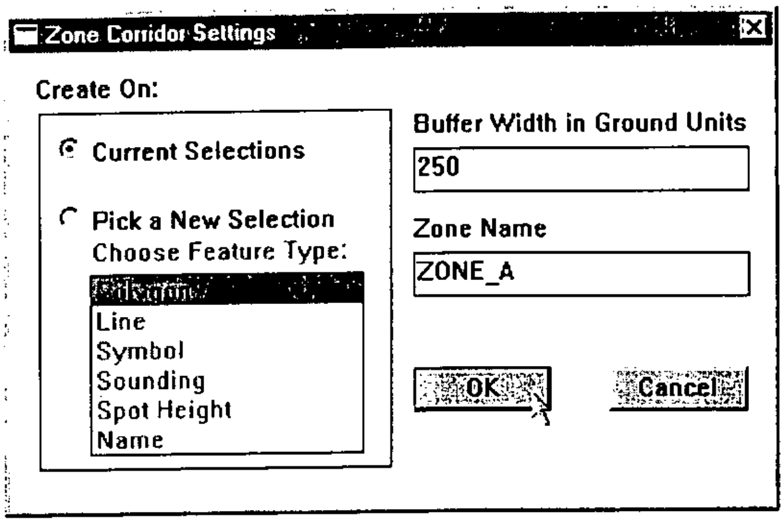

Pixel Resolution Settings

User Defined Resolution

Value (Ground Units): 5

Figure 4.17 Pixel Resolution Settings Dialog Box.

10. Click Utilities>Zone>Create>By Corridor to open the Zone Corridor Settings Dialog Box. Click the current selections option, enter buffer width and zone name, and then click **OK** to dismiss the box. A zone will be created and displayed on the map canvas as shown in Figure 4.19.


Zone Corridor Settings

Create On:

(●) Current Selections

( ) Pick a New Selection

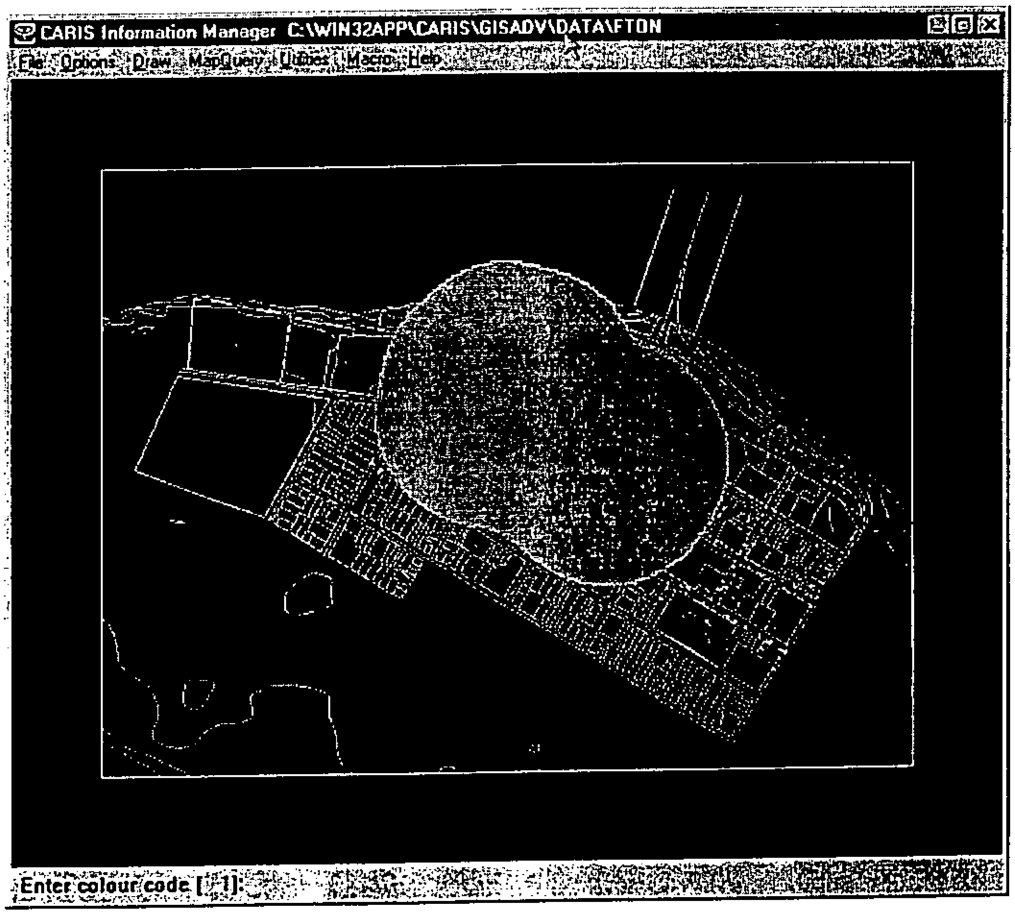
Choose Feature Type:

Polygon
Line
Symbol
Sounding
Spot Height
Name

Buffer Width in Ground Units: 250

Zone Name: ZONE_A

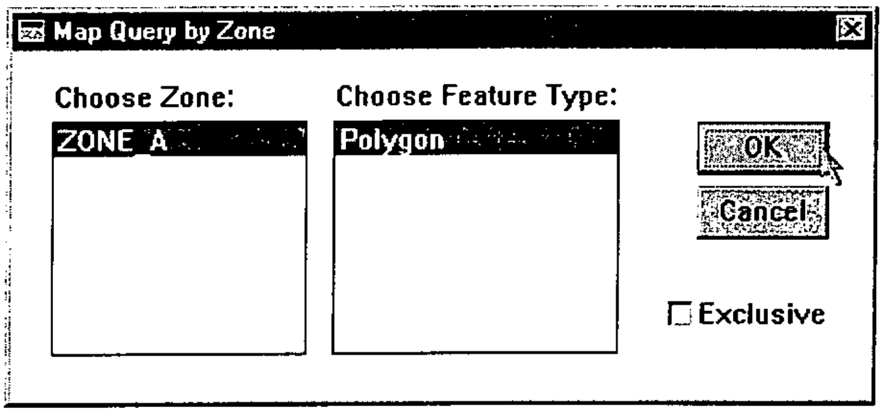
Figure 4.18 Enter values in the Zone Corridor Settings Dialog Box.


CARIS Information Manager C:\WIN32APP\CARIS\GISADV\DATA\FTON

Enter colour code [-1]:

Figure 4.19 Zone Created and Displayed on the Map Canvas.

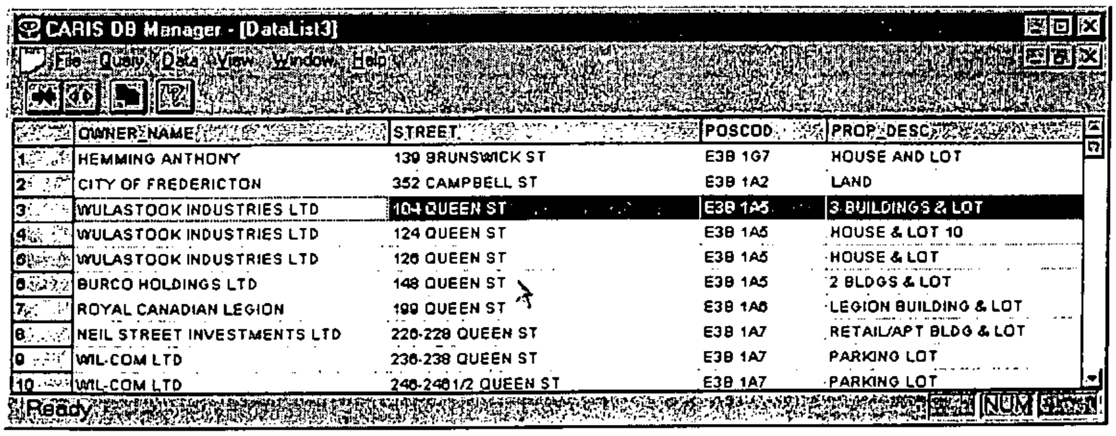
11. Click Map Query>By Zone to display the following dialog box. In the Choose Zone box, select ZONE_A, and in the Choose Feature Type, select Polygon. Then Click OK to execute the operation. The result will be displayed in CARIS DB Manager as shown in Figure 4.21.



Choose Zone: ZONE_A

Choose Feature Type: Polygon

Figure 4.20 Map Query by Zone.


CARIS DB Manager - [DataList3]
File Query Data View Window Help
Ready
NUM

	OWNER_NAME	STREET	POSCOD	PROP_DESC
1	HEMMING ANTHONY	139 BRUNSWICK ST	E3B 1G7	HOUSE AND LOT
2	CITY OF FREDERICTON	352 CAMPBELL ST	E3B 1A2	LAND
3	WULASTOOK INDUSTRIES LTD	104 QUEEN ST	E3B 1A5	3 BUILDINGS & LOT
4	WULASTOOK INDUSTRIES LTD	124 QUEEN ST	E3B 1A5	HOUSE & LOT 10
5	WULASTOOK INDUSTRIES LTD	126 QUEEN ST	E3B 1A5	HOUSE & LOT
6	BURCO HOLDINGS LTD	148 QUEEN ST	E3B 1A5	2 BLDGS & LOT
7	ROYAL CANADIAN LEGION	199 QUEEN ST	E3B 1A6	LEGION BUILDING & LOT
8	NEIL STREET INVESTMENTS LTD	226-228 QUEEN ST	E3B 1A7	RETAIL/APT BLDG & LOT
9	WIL-COM LTD	236-238 QUEEN ST	E3B 1A7	PARKING LOT
10	WIL-COM LTD	246-248 1/2 QUEEN ST	E3B 1A7	PARKING LOT

Figure 4.21 Query Results of the Map Query by Zone operation.

Finally, you got the answer to the spatial question. This illustrates an example of how to use the application description to assist users performing spatial analysis.

4.6 Limitations

According to the current implementation, although it can demonstrate a viable usage of the application model, it has the following limitations:

1. The current implementation does not provide any automatic certification on the content of input application model. It is because the application model only provide a schema, which helps users to determine what is required to be entered, while it is impossible to check if the content is viable or not. Therefore, one who designs the application should check the application usability before it can be stored in the data store.

2. There is a metadata feature that helps users to determine keywords for searching data set. However, the keywords provided are only an initial guess, it does not guarantee if it can hit a best-fit data set from the data clearinghouse.

3. The current implementation bases on the application model that described in Chapter 3. It focuses on modeling geospatial application information. However, in practical geospatial analysis, it may involve non-geospatial application that does not cover in the developed model. To extent the application model capability, we can make it much generic, while the

generic model may further derive into different special themes such as geospatial application or other special applications.

4.7 Summary

In this chapter, we went through the implementation of the geospatial application information browser, the establishment of Z39.50 connection, an example to demonstrate an application scenario and limitations of current implementation. With the browser, it is recognized that it can help users to learn and retrieve application information. If users interest to particular spatial data, a metadata searching facility could help to retrieve spatial metadata from the geospatial data clearinghouse. The technology that we use in building the application includes XML, DOM and Z39.50. We use Java for the implementation and divide the entire application into three packages for easier handling. In addition, a testing suite was setup for testing the application and Z39.50 connection.

CHATPER 5
CONCLUSIONS AND RECOMMENDATIONS

The motivation of this study is to help users determine appropriate geospatial resources and operations for solving geospatial analysis and modeling problems. We have designed and developed a geospatial application model or simply an application model, based on the concept of geospatial metadata for providing the solution and this is the objective of this study. A system for documenting geospatial data processing flow is then developed based on the idea.

In the beginning of this study, the current development for creating, storing and distributing geospatial metadata was thoroughly reviewed, while the geospatial metadata management paradigm was introduced to provide a clear picture for understanding the metadata management in the GIS community. The paradigm contains three components including the metadata standard, the metadata management system and the metadata information system. The motivation for building geospatial metadata standard is to facilitate understanding and exchange of geospatial data. Consequently, it helps users to find and identify spatial data fitting to their own application requirements. Based on this concept, the application model for describing geospatial procedures was developed. With the simple and systematic structure, the application model provides a set of easy guidelines for understanding and documenting application procedure. Nevertheless, the application model is not an individual information element; it resides on and is part of the data clearinghouse. In fact, the model requires metadata document to provide a completed description for a geospatial application. This is reasonable and logical, since there is no geospatial application without input data. Of course, geospatial application also generates new data set, no matter it is in any data format and usually this forms a chain between applications and data sets.

To prove the concept, prototype of a geospatial application management browser was built, which could be used to view the application information with content based on the geospatial application model and search metadata document from a geospatial data clearinghouse. XML (Extensible Markup Language) was used to build the information structure and data, while using DOM (Document Object Model) to provide program objects for manipulating the information. In addition, a Z39.50 interface to the system was incorporate into the system, which allows users to search and retrieve geospatial metadata directly from Z39.50 server or geospatial data clearinghouse server.

To perform testing of the prototype and its connection with data clearinghouse via Z39.50 protocol, a testing suite was setup with an Isite server, an Apache web server and an HTTP-Z39.50 gateway running on Linux. Meanwhile, the prototype can run on any machine i.e. either PC or UNIX, providing that it equips with Java virtual machine. Furthermore, all the testing software is running in TCP/IP network environment.

This testing suite allows confirming that the geospatial application information browser can provide capability to help users to learn and understand the available application procedures. However, it is necessary to further improve the browser, so that it can provide better capability and facility for practical use.

5.1 Conclusions

The study demonstrated the analysis, design and implementation of the application model for describing geospatial application. This application model enables geospatial application designer to write application concept and procedure for solving particular spatial problem. To be clear, the application model is not designed for delivering a spatial model for solving particular spatial problem. Instead, it provides structure and guidelines, which forms a framework for building and documenting application information. The application model has a tree structure and XML is used to implement it. Therefore, the application model itself is readable, understandable by both machine and human. These are, in fact, benefits inherited from the XML. By implementing the model using XML, it helps to create document for describing geospatial application. XML is an easy to learn and understand document definition system for building custom markup language. XML document is self-descriptive, in which each data element contains a pair of well-defined tags that tell the attribute type, and a data element that tells the attribute value.

With the application information wrapped up by XML, we use DOM to build program for manipulating, organizing any geospatial application information contents. In this sense, the XML files, which contains application information similar to a data store, while the DOM is similar to but powerful than a data manipulation language (DML) in ordinary database system. Therefore, the application can manipulate and show the information properly with intuitive graphic user interface. It can present the application information in tree view, so that by consequence expanding the branches of tree nodes, more related information will be shown accordingly. The benefit of the application model is thus cleared, which provides a structure for organizing geospatial knowledge.

In addition, in the current implementation, flat files were used to store the descriptions. There are two types of description: Application description and SpatialData description. The combination of them forms an application model. The reason to split the model into two parts is to remove data duplication, since the spatial data is a repeatable data element in various applications. Again, it is part of the Application Model and it will not be used independently.

Current implementation could be done with better performance and data management capability with database system such as RDBMS and OODBMS. The prototype built in the study for storing application description can only handle a few data files. If there are too many files to be opened, the performance will be degraded severely. The degree of performance degrading highly relies on available memory on the machine.

Nevertheless, the description can definitely provide information for experienced users to understand those available spatial analysis techniques. For novice GIS users, the description also provides ready to use information for performing an analysis without starting from scratch, which facilitates reusability of geospatial information. In addition, if users would like to retrieve related geospatial metadata, a Z39.50 connection can be built with any data clearinghouse server for this purpose. The study provides a solution for building a framework for describing geospatial application in any type. This in fact meets the objective in the study that helps users to find geospatial resources for solving spatial analysis and modeling problems.

5.2 Recommendations

The objective of recommendations is to suggest modifications for the Application Model and the geospatial application information browser to make it better and more efficient in data management and distribution as well as advancement in system capability. Since the prototype is an experimental implementation of the application model, it is necessary to be improved and modified the application program in various aspects for practical use.

The main component of the prototype is the data store. In the current implementation, file system is used for managing data files, which is not sufficient for building as a data management and storage system in practical sense. As examined different types of database system in the market such as flat files, relation database system, and object-oriented database system, it is concluded that the most suitable one is the object-oriented database system. It is because the tree structure of the XML document is well fitted to the object model without any less of semantics, while the application model is implemented on XML. In addition, the object-oriented database can provide better efficiency on data retrieval compared to other methods. However, the reason why flat files was used is simply because it had no extra cost and was easier to implement.

The Application Model provides a basic concept for documenting geospatial-processing procedure. There are process logic and procedures. The process logic provides the information flow and concept to solve particular spatial problem. Meanwhile, the procedure provides information about practical GIS commands that apply on real data set with spatial operators and operational sequences. Although there is a set of operating tool provided via the implementation for manipulating the information, the information contents is not flexible enough to be reused. This means, a geospatial application could be documented and reused as a whole. In fact, a geospatial application contains multiple modular tasks. Different combination of modular tasks could provide different application for performing analysis. Therefore, an additional set of management tool should be introduced to provide relax on the flexibility for reusing the application model. The goal for this improvement enables the reuse of modular tasks for different geospatial application, instead of the reuse of entire geospatial application in the current case.

In addition, descriptive items developed for GUI operation could be further studied. GUI operation is a complex task different from the straightforward command line operation. As observed from the GUI operation for performing particular geospatial application, it may involve various GUI components and this is usually adopted by newly developed GIS software. A language definition could be developed, which help to specify for describing GUI and its operation, in turn, this helps to enhance the Application Model.

To encourage GIS users to publish their own application procedure is another significant issue for enabling the idea of sharing application knowledge. We need to enrich the information content of an application store. To accomplish this, a mechanism for different users to upload application information is required. In this case, it should provide a validation module for users to check syntax and grammar of their own documents before putting it into the database.

Relating to the current system architecture, the application model resides on the client side and it is implemented as a Java application program. This is inflexible for Internet users, but it is easier to implement for testing purpose. In fact, the application program can be converted into Java applet. Java applet is a small-size program, which requires a Java virtual machine to run it properly on client machine. The benefit of using Java applet is that, it allows any Internet users who equip with a Java complaint browser to use the program. In addition, it does not require any explicit installation and will do the installation automatically once the user's browser reaches the web page that contains Java applet in the Internet. It allows users to access and use it without any cumbersome procedure.

Besides, the application model can reside on the server side instead of the client side, which puts all the processing loading to the server side and minimize the loading on client side. Of course, it is expected that the server machine would have better processing power and resources when comparing with the client machine. The application model on the server side enables monitoring of client activities and the centralized approach for storing the application information can provide easier and better management of data.

Another possible architecture for implementing the application model is to incorporate it into GIS. The idea of this is called metadata integration, which is a technology for enabling change detection and automatic metadata updating of database. It is recognized that geospatial metadata is part of the geospatial data stored in the geospatial database, maintenance of these is a responsibility of the GIS. If this can be done, data users can acquire up-to-date metadata, while data producers can reduce cost and time spent for metadata maintaining. Consequently, providing the metadata content is sufficient enough, further exploitation of metadata can be done to determine the changing trend of the underlying geospatial database over time. In turn, this is possible to help answering questions such as where is the most active changing area, how does the accuracy change of results after analysis and so on.

Furthermore, to ensure data sharing and usage of information, a standard is the only way to provide a common language for information exchanging and communicating. Similar to CSDGM, the final goal of the application model should be a standardization of the application description among a large group of GIS users. Therefore, anyone who builds procedure for solving particular spatial problem could apply the standard to document it. The benefit of this act is obvious, which helps followers to reuse application procedure with minimize efforts. Nevertheless, it is a long-term development with massive resource and not an easy task for implementation.

Ultimately, with the implication of the application model, the current implementation and recommendations, it can further develop and build a geospatial application management system as well as a geospatial analysis compiler.

It is expected that the geospatial application management system could offer two levels of service and works in the Internet. The first level service provides typical data management services for users to store and retrieve application knowledge. The second level service provides advanced services such as ready-to-run spatial analysis and modeling procedure, metadata integration to existing GIS as well as interface to spatial data clearinghouses for accessing metadata documents. Eventually, this management system could be a practical and more realistic implementation for helping GIS users sharing and reusing geospatial-processing knowledge.

For the geospatial analysis compiler, it could be an extension of the application model. This compiler can process statements written in form of pseudo code like language and turns them into another set of statements that is used for particular GIS or even machine code that is used for computer's processor. If the compiler supports generation of GIS commands, it is expected that the geospatial analysis compiler should understand various GIS software, and hence by writing pseudo code once, it can run on any supported GIS software. Similarly, for the second case which generates machine code, this is much advanced and flexible for performing spatial analysis, since the code generated is independently with GIS software and expects to run on any supporting operating system. For both cases, these require a comprehensive study on various GIS software and creation of a simple and easy to understand programming language dedicated for writing statements with geospatial sense and operations.

Finally, it is believed that the concept of geospatial metadata management and the application model developed in this study is a new trend and will play a significant role in future GIS.

REFERENCES

ANSI/NISO (1995). "Information Retrieval (Z39.50): Application Service Definition and Protocol Specification. ANSI/NISO Z39.50-1995." Z39.50 Maintenance Agency, U.S.A. URL: http://lcweb.loc.gov/z3950/agency.

Apparao, V., S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A.L. Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood (1998). "Document Object Model (DOM) Level 1 Specification Version 1.0." W3C Recommendation, W3C. URL: http://www.w3c.org/TR/1998/REC-DOM-Level-1-19981001.

Australia New Zealand Land Information Council (1997). "Metadata Working Group." URL: http://www.anzlic.org.au/metagrp.htm.

Bernus, P., and G. Schmidt (1998). "Architectures of Information Systems." In Handbook on Architectures of Information Systems. Bernus, P., K. Mertins, and G. Schmidt, Springer, Berlin, Germany.

Bray, T., J. Paoli, and C.M. Sperberg-McQueen (1998). "Extensible Markup Language (XML) 1.0." W3C Recommendation, W3C URL: http://www.w3c.org/TR/1988/REC-xml-19980210.

Comité Européen de Normalisation (1997). "CEN/TC 287 Overview." URL: http://forum.afnor.fr/afnor/WORK/AFNOR/GPN2/Z13C/PUBLIC/DOC/287n540 .doc.

Center for Networked Information Discovery and Retrieval (1997). "Isite Information System." URL: http://www.cnidr.org/ir/isite.html.

Clark, J. (1997). "Comparison of SGML and XML." W3C Note, W3C. URL: http://www.w3.org/ NOTE-sgml-xml-971215.

Cornell, G., and C.S. Horstmann (1998). Core Java (3rd Edition). SunSoft Press/Prentice Hall, Upper Saddle River, New Jersey, U.S.A.

Douglas E.C. (1997). Computer Networks and Internets. Prentice Hall, Upper Saddle River, New Jersey, U.S.A.

Domaratz, M. (1995) "Finding and Accessing Spatial Data in the National Spatial Data Infrastructure" Geographic Information Systems and Libraries: Patrons, Maps, and Spatial Information, pp.31-40.

Environment Science Research Institute Inc (1997). "Document.aml Version 7.1.1." Redlands, Calif. U.S.A., URL: http://www.esri.com/base/products/arcinfo/docaml711.html.

Federal Geographic Data Committee (1994a). "Executive Order 12906, COORDINATING GEOGRAPHIC DATA ACQUISITION AND ACCESS: THE NATIONAL SPATIAL DATA INFRASTRUCTURE." USGS, Reston, VA, U.S.A., URL: http://www.fgdc.gov/publications/documents/geninfo/execord.html.

Federal Geographic Data Committee (1994b). "Content Standards for Digital Geospatial Metadata, June 8, 1994." USGS, Reston, VA, U.S.A., URL: ftp://www.fgdc.gov/pub/metadata/meta6894.

Federal Geographic Data Committee (1997a). Framework Introduction and Guide. FGDC, USGS, Reston, VA, U.S.A.

Federal Geographic Data Committee (1997b). "Federal Geographic Data Committee" USGS, Reston, VA, U.S.A.

Gamiel, K. (1994). "The Isite Information System. Version 1.05." Publication of the MCNC, Clearinghouse for Networked Information Discovery and Retrieval, North Carolina, U.S.A.

Harlan J.O., and G. Rushton (1995). Sharing Geographic Information. Center for Urban Policy Research, Rutgers University.

Hart, D., and H. Philips (1997). "Metadata Primer – A "How to" Guide on Metadata Implementation" National States Geographic Information Council, U.S.A. URL: http://localis2.lic.wisc.edu/~dhart/metaprim.htm.

Fielding, R., J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee (1999). "Hypertext Transfer Protocol -- HTTP/1.1." INTERNET-DRAFT, HTTP Working Group , Internet Engineering Task Force (IETF). URL: http://www.w3.org/Protocols/HTTP/1.1/draft-ietf-http-v11-spec-rev-06.txt.

IBM (1998). "IBM's XML Parser for Java." IBM Research. New York, U.S.A. URL: http://www.alphaworks.ibm.com/tech/XML4J.

Illinois Natural Resources Geospatial Data Clearinghouse (1997). "FGDCMETA.AML." Illinois State Geological Survey, U.S.A. URL: http://www.isgs.uiuc.edu/nsdihome/webdocs/fgdcmeta.html.

Intergraph Corporation (1994). MGE Analyst Reference Guide. Intergraph Corporation, Huntsville, Alabama, U.S.A.

International Organization for Standardization (1997). "ISO/TC211/WG3 Geospatial data administration." URL: http://www.statkart.no/isotc211/wg3/wg3welc.htm.

Koch, G., and K. Loney (1997). ORACLE: The Complete Reference. Osborne McGraw-Hill, New York, U.S.A.

Korth, H.F., and A. Silberschatz (1991). Database System Concepts. 2nd rev. ed., McGraw-Hill, New York, U.S.A.

Lower Colorado River Authority (1997). "Metadata Management System." URL: http://www.tnris.state.tx.us/lcracompiler.htm.

Lee, Y.C. (1995). GIS for the Curious. Geomatics Canada.

Lee, Y.C., and H.C. Chan (1998). Report on Metadata Creation for Geology Digital Map. Unpublished Consultancy Report (Agreement No. CE42/95) for the Geotechnical Engineering Office, Civil Engineering Department, Hong Kong Special Administrative Region Government, China, 54 pp.

LeVan, R. (1990). "Building A Z39.50 Client." OCLC Online Computer Library Center Inc. URL: ftp://ftp.rsch.oclc.org/pub/SiteSearch/z39.50_client_api/zclient.doc.

Maruyama, H., N. Uramoto, and K. Tamura (1998). "The Power of XML: XML's Purpose and Use in Web Applications." IBM Research, New York, U.S.A. URL: www.ibm.com/xml.

McGrath, S. (1998). ParseMe.1st, SGML for software developers. Prentice Hall, Upper Saddle River, New Jersey, U.S.A.

National Biological Information Infrastructure (1997). "NBII MetaMaker Version 2.10." U.S. Geological Survey. URL: http://www.emtc.nbs.gov/http_data/emtc_spatial/applications/nbiimker.html.

Nebert, D.D (1997). "WWW Mapping in a Distributed Environment, Scenario of visualizing mixed remote data." Distributed Mapping Scenario for the WWW Mapping Special Interest Group of the OpenGIS Consortium.

Nebert, D.D. (1999). "Z39.50 Application Profile for the Content Specification for Digital Geospatial Metadata or "GEO", Version 2.2." FGDC, US Geological Survey, Reston, Virginia, U.S.A.

URL:http://www.blueangeltech.com/Standards/GeoProfile/geo22.htm.

Nebert, D.D. and J. Fulton (1995b). "Use of Z39.50 to Search and Retrieve Geospatial Data.", Digital Libraries Conference, June 1995.

Open GIS Consortium (1998). "The OpenGIS Specification Model, Topic 11: Metadata Version 3.1." OpenGIS® Abstract Specification, Open GIS Consortium.

Peng, Z.R., and D.D. Nebert (1997). "An Internet-based GIS Data Access System." Proceeding of GIS AM/FM ASIA 97 & GeoInformatics '97, Taipei, Taiwan, May 26-29, 1997, pp.431-445.

Raggett, D (1997). "HTML 3.2 Reference Specification." W3C Recommendation. URL: http://www.w3.org/TR/REC-html32/.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen (1991). Object-oriented Modeling and Design. Prentice Hall, Englewood Cliffs, New Jersey, U.S.A.

Tomlin, C.D. (1990). Geographic Information Systems and Cartographic Modeling. Prentice Hall, Englewood Cliffs, New Jersey, U.S.A.

U.S. Corps of Engineer (1997). "Geospatial Data Clearinghouse Node." U.S.A. URL: http://corpsgeo1.usace.army.mil/

U.S. Geological Survey (1997). "Formal Metadata – info and tools." National Spatial Data Infrastructure, U.S.A. URL: http://geochange.er.usgs.gov/pub/tools/metadata/

U.S. Geological Survey (1999a). "A Compiler for Formal Metadata." USGS, U.S.A. URL:http://geology.usgs.gov/tools/metadata/tools/doc/mp.html.

U.S. Geological Survey (1999b). "A Pre-parser for Formal Metadata." USGS, U.S.A. URL: http://geology.usgs.gov/tools/metadata/tools/doc/cns.html.

World Wide Web Consortium (1998). “Extensible Markup Language (XML^{TM}) Activity” W3C Architecture Domain, Activity Statement. URL: http://www.w3c.org/XML/Activity.html.

BIBLIOGRAPHY

Blaha, M., and W. Premerlani (1998). Object-oriented Modeling and Design for Database Applications. Prentice-Hall, Upper Saddle River, New Jersey, U.S.A.

Callahan, E. (1997). Microsoft Access 97/Visual Basic step by step. Microsoft Press, Redmond, Washington, U.S.A.

Chan, E.H.C., and Y.C. Lee (1999). "Geospatial Metadata and its Management" *The International Symposium on Spatial Data Quality* (ISSDQ '99). Hong Kong, 18-20 July. The Hong Kong Polytechnic University, Hong Kong Special Administrative Region, China, pp. 170-177.

Chan, E.H.C., and Y.C. Lee (1999). "Application Base Management System." *The Second International Workshop on Dynamic and Multi-Dimensional GIS* (DMGIS'99). J. Chen, Q. Zhou, Z. Li, and J. Jiang (Eds.) Beijing, 4-6 October. ISPRS and IGU Study Group, Beijing, China, pp. 167-172.

Coad, P., and M. Mayfield (1996). Java Design: Building Better Apps & Applets. Prentice Hall, Upper Saddle River, New Jersey, U.S.A.

Graham, I. (1991). Object Oriented Methods. 2nd rev. ed., Addison-Wesley, Workingham, England.

Gray, P.M.D., K.G. Kulkarni., and N.W. Paton (1992). Object-Oriented Databases: A Semantic Data Model Approach. Prentice Hall, Upper Saddle River, New Jersey, U.S.A.

Gobel, S., and K. Lutze (1998). "Development of meta databases for geospatial data in the WWW." Proceedings of the 6th international symposium on Advances in Geographic Information Systems, ACM-GIS '98, Washington, DC, U.S.A., 6-7 November., pp. 94-99.

Hart, D., and H. Phillips (1997). "Metadata Primer -- A "How To" Guide on Metadata Implementation." National States Geographic Information Council, U.S. URL: http://www.lic.wisc.edu/metadata/metaprim.htm.

Intergraph Corporation (1994). MGE Basic Administrator Reference Guide. Intergraph Corporation, Huntsville, Alabama, U.S.A.

Intergraph Corporation (1994). MGE Basic Nucleus Reference Guide. Intergraph Corporation, Huntsville, Alabama, U.S.A.

Sun Microsystems, Inc. (1997). "JDK 1.1.6 Documentation." Sun Microsystems, Inc., Palo Alto, CA, U.S.A.

Sun Microsystems, Inc. (1999). "Java® 2 Platform API Specification." Sun Microsystems, Inc., Palo Alto, CA, U.S.A.

Kim, W. (1990). Introduction to object-oriented databases. MIT Press, Cambridge, Mass., U.S.A.

Lang, U., G. Grinstein, and R.D. Bergeron (1995). "Visualization Related Metadata." Proceedings of Database issues for data visualization: IEEE Visualization '95 Workshop, Atlanta, Georgia, U.S.A. October, 1995, pp.26-27.

Lee, T.B. (1997). "Web Architecture: Metadata." World Wide Web Consortium URL: http://www.w3.org/DesignIssues/Metadata.html.

Library of Congress (1998). "Z39.50 Maintenance Agency." Library of Congress, U.S.A. URL: http://lcweb.loc.gov/z3950/agency/.

Light, R. (1998). Presenting XML. Sams.net Publishing, Indianapolis, IN, U.S.A.

Nairn, A., and B. Irwin (1997). "The Australian Spatial Data Infrastructure: Its current status and future direction." Cartography, Vol. 26, No.1, June 1997.

National Biological Information Infrastructure (1997). "Building the Infrastructure: Standards, Protocols, Tools Relating to the NBII, and Applications." National Biological Information Infrastructure, U.S. Geological Survey. URL: http://www.nbii.gov/infrastructure/.

Microsoft Corporation (1999) "Virtual Private Networking: An Overview", Microsoft, Redmond, Washington, U.S.A. URL: www.msdn.microsoft.com/workshop/server/feature/vpnovw.asp.

Nebert, D.D. (1997). "Metadata Tools Survey." Federal Geographic Data Committee, U.S.A. URL: http://www.fgdc.gov/Metadata/Toollist/metatools797.html.

Object Design, Inc. (1998). "An XML Data Server For Building Enterprise Web Applications" Object Design, Inc. Burlington, MA, U.S.A.

Object Design, Inc. (1998). "ObjectStore PSE/PSE Pro for Java API User Guide, Release 1.1." Object Design, Inc. Burlington, MA, U.S.A.

Object Design, Inc. (1998). "ObjectStore PSE/PSE Pro for Java Tutorial, Release 2.0." Object Design, Inc. Burlington, MA, U.S.A.

Phillips, H. (1997). "Metadata Tools for Geospatial Data." The Wisconsin Land Information Clearinghouse, U.S. URL: http://badger.state.wi.us/agencies/wlib/sco/metatool/mtools.htm.

Red Hat Software, Inc. (1998). Red Hat Linux 5.2: The Official Red Hat Linux Installation Guide. Red Hat Software, Inc., Research Triangle Park, North Carolina, U.S.A.

Reese, G. (1998). Database Programming with JDBC and JAVA. O'Reilly, Sebastopol, CA, U.S.A.

Ricart, M.A. (1996). Apache Server survival guide. Sams.net Publishing, Indianapolis, IN, U.S.A.

Royappa, A.V (1999). "Implementing Catalog Clearinghouse with XML and XSL." Proceedings of the 1999 ACM Symposium on Applied Computing 1999, Feb. 28-March 2, 1999, San Antonio, TX, U.S.A., pp.616-623.

Stitt, S. (1997). The FGDC Content Standard for Digital Geospatial Metadata: An Image Map. Center for Biological Informatics, U.S. Geological Survey. URL: http://biology.usgs.gov/nbs/meta/meta.htm.

Tomlin, C.D. (1983). Digital Cartographic Modeling Techniques in Environmental Planning. Ph.D dissertation, Yale University, U.S.A., 290 pp.

Weske, M., G. Vossen, C. B. Medeiros, and F. Pires (1998). "Workflow Management in Geoprocessing Applications." *Proceedings of the Sixth International Symposium on Advances in Geographic Information Systems*, Nov. 6-7 1998, Washington, D.C., U.S.A., pp. 88-94.

APPENDIX I - DOCUMENT TYPE DEFINITION (DTD) AND SAMPLE XML DATA FILES

The following are DTDs and a set of sample data for application information and spatial data information.

DTD for the Application Information

```
<?xml encoding="US-ASCII"?>

<!--Revision: 0.1 applicationX.dtd

   This is a DTD for describing geospatial processing procedure language.
   This document type definition (DTD) is developed Department of
   Land Surveying & Geo-Informatics, The Hong Kong Polytechnic University
   under research topic "Develop an Application Model for Geospatial Data
   Clearinghouse" All rights reserved 1998-1999 -->

<!--Declare the parameter entities-->
<!ENTITY % id.global "ID ID #REQUIRED">

<!ELEMENT application_model (application+)>

<!--This is main content of the application model language which
   describes an application as three components including
   a description, a pseudecode and a series of procedure -->
<!ELEMENT application (project?, name, description, pseudocode, procedure*)>
<!ATTLIST application
          %id.global;
          CATEGORY ( SPATIAL | GRID | NETWORK | TERRAIN | IMAGE |MIXED )
"SPATIAL">

<!--Project element helps to provide and organize project information
     in a large scale GIS project which may involve multiple type of
     GIS analysis procedures.-->
<!ELEMENT project (title, subtask_name, organization, date?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT subtask_name (#PCDATA)>
<!ELEMENT organization (#PCDATA)>
<!ELEMENT date (day?, month, year)>
<!ELEMENT day (#PCDATA)>
<!ELEMENT month (#PCDATA)>
<!ELEMENT year (#PCDATA)>

<!ELEMENT description  (#PCDATA)>

<!ELEMENT pseudocode (list)+>

<!ELEMENT list (#PCDATA)>

<!--Either shell command script or gui operation can be entered.
```

```
 Hence, it does not permit a procedure to enter both method at a time.-->
<!ELEMENT procedure  (gis,gis_version,(shell | gui),result?)>
<!ATTLIST procedure
          NAME CDATA #IMPLIED>

<!ELEMENT gis (#PCDATA)>
<!ELEMENT gis_version (#PCDATA)>

<!ELEMENT gui (step)+>
<!ELEMENT step (menu_selection, panel_entry)+>
<!ATTLIST step
          %id.global;>

<!ELEMENT menu_selection (#PCDATA)>

<!ELEMENT panel_entry (title,(button_action|variable)*)>

<!ELEMENT button_action (#PCDATA | variable)*>
<!ATTLIST button_action
          TYPE (single_click | double_click | select) "single_click">

<!ELEMENT shell (statement)+>
<!ATTLIST shell
          TYPE CDATA #REQUIRED>
<!ELEMENT statement (#PCDATA | variable)*>

<!ELEMENT result (variable)+>

<!ELEMENT variable (name,value)>
<!ATTLIST variable
          LINK CDATA #IMPLIED
          TYPE (spatial | general) "spatial"
          DESC CDATA #IMPLIED
          FILENAME CDATA #IMPLIED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

DTD for the Spatial Data Information

```
<?xml encoding="US-ASCII"?>
<!--Revision: 0 spatialdata.dtd, xml4j,_1_1_4-->
<!ELEMENT spatialdata (idinfo)+>
<!ELEMENT idinfo (name, descript, bounding?, keywords, from_application?)>
<!ATTLIST idinfo ID ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT descript (#PCDATA)>
<!ELEMENT bounding (westbc, eastbc, northbc, southbc)>
<!ELEMENT westbc (#PCDATA)>
<!ELEMENT eastbc (#PCDATA)>
<!ELEMENT northbc (#PCDATA)>
<!ELEMENT southbc (#PCDATA)>
<!ELEMENT keywords (theme, place?, stratum?, temporal?)>
```

```
<!ELEMENT theme (themekey+)>
<!ELEMENT place (placekey+)>
<!ELEMENT stratum (stratkey+)>
<!ELEMENT temporal (tempkey+)>
<!ELEMENT themekey (#PCDATA)>
<!ELEMENT placekey (#PCDATA)>
<!ELEMENT stratkey (#PCDATA)>
<!ELEMENT tempkey (#PCDATA)>
<!ELEMENT from_application (#PCDATA)>
<!ATTLIST from_application
        application_ID CDATA #REQUIRED>
```

Sample XML data for the Application information

```
<?xml version="1.0"?>
<!-- Revision: 0 java/Applicaton/applicationX.xml, xml4j_1_1_9 -->
<!DOCTYPE application_model SYSTEM "applicationX.dtd">
<application_model>
 <application ID="UGTL1" CATEGORY="SPATIAL">
  <name>Underground gas tank leakage (First Approach)</name>
  <description>Finding affected parcels if the gasoline tanks in all gas stations leak at the same
time</description>
  <pseudocode>
        <list>1.Retrieve from the database all gas stations.</list>
        <list>2.Generate buffer zones around them to simulate the area affected by a leakage.</list>
        <list>3.Overlay the zones and the parcels to find the affected parcels.</list>
  </pseudocode>
  <procedure>
   <gis>CARIS for Windows</gis>
   <gis_version>1.0</gis_version>
   <gui>
    <step ID="step1">
        <menu_selection>CARIS Information Manager>Open</menu_selection>
        <panel_entry>
                <title>Enter name of CARIS file to open</title>
                <variable LINK="BASEMAP" TYPE="spatial" DESC="A data set contains land
parcels, gas station">
                        <name>File Name</name>
                        <value>fton.des</value>
                </variable>
        </panel_entry>
   </step>
   <step ID="step2">
   <menu_selection>CARIS Database Manager>Query>Query Condition</menu_selection>
        <panel_entry>
                <title>Query Condition"</title>
                <variable TYPE="general" DESC="Table name">
                        <name>Name</name>
                        <value>PROPERTY</value>
                </variable>
                <variable TYPE="general" DESC="Table column">
                        <name>Column</name>
```

```
                    <value>PROP_TYPE</value>
               </variable>
               <variable TYPE="general" DESC="Logical operator">
                    <name>Operator</name>
                    <value>=</value>
               </variable>
               <variable TYPE="general" DESC="Search value, gas station">
                    <name>Value</name>
                    <value>203</value>
               </variable>
               <button_action TYPE="single_click">Add</button_action>
               <button_action TYPE="single_click">OK</button_action>
          </panel_entry>
     </step>
     <step ID="step3">
     <menu_selection>CARIS Information Manager>Options>Resolution</menu_selection>
          <panel_entry>
               <title>Pixel Resolution Settings</title>
               <variable TYPE="general" DESC="Ground resolution">
                    <name>User defined resolution value (Ground Units)</name>
                    <value>5</value>
               </variable>
               <button_action TYPE="single_click">OK</button_action>
          </panel_entry>
     </step>
     <step ID="step4">
     <menu_selection>CARIS Information Manager>Zone>Create>By Corridor</menu_selection>
          <panel_entry>
               <title>Zone Corridor Settings</title>
               <button_action TYPE="select">Current Selection</button_action>
               <variable TYPE="general" DESC="Buffer width">
                    <name>Buffer Width in Ground Units</name>
                    <value>250</value>
               </variable>
               <variable TYPE="spatial" DESC="Zone Name">
                    <name>Zone Name</name>
                    <value>ZONE_A</value>
               </variable>
               <button_action TYPE="single_click">OK</button_action>
          </panel_entry>
     </step>
     <step ID="step5">
     <menu_selection>CARIS Information Manager>MapQuery>By Zone</menu_selection>
          <panel_entry>
               <title>Map Query by Zone</title>
               <button_action TYPE="select">
               <variable TYPE="spatial" DESC="Zone">
                    <name>Choose Zone</name>
                    <value>ZONE_A</value>
               </variable>
               </button_action>
               <button_action TYPE="select">
               <variable TYPE="general" DESC="Feature type">
                    <name>Choose Feature Type</name>
                    <value>Polygon</value>
               </variable>
```

```
                    </button_action>
                    <button_action TYPE="single_click">OK</button_action>
            </panel_entry>
    </step>
   </gui>
  </procedure>
 </application>
 <application ID="Digi">
  <name>Manual Digitizing from hardcopy maps</name>
  <description>How to build a basemap</description>
  <pseudocode>
          <list>1.Prepare a paper map.</list>
          <list>2.Put the map on a digitizer.</list>
          <list>3.Register the map using known points and coordinates to the digitizer.</list>
          <list>4.According to selected feature, follow the line or shape of the map, digitize the map into
the computer.</list>
  </pseudocode>
  </application>
</application_model>
```

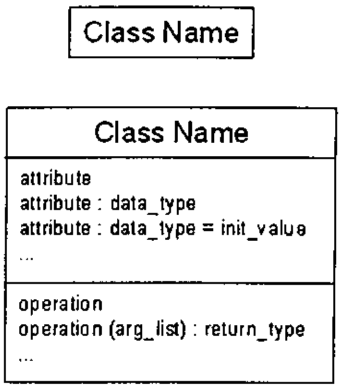
Sample XML data for the Spatial Data Information

```
<?xml version="1.0"?>
<!-- Revision: 0 java/applicaton/spatialdata.xml, xml4j_1_1_4 -->
<!DOCTYPE spatialdata SYSTEM "spatialdata.dtd">
<spatialdata>
<idinfo ID="UGTLD1">
        <name>Underground Gas Tank Leakage Analysis Result (First Approach)</name>
        <descript>A data set contains affected parcels after analysis using the underground gas tank
leakage (First Approach)</descript>
        <keywords>
                <theme>
                        <themekey>basemap</themekey>
                        <themekey>affected parcels</themekey>
                        <themekey>gas tank leakage</themekey>
                        <themekey>gas leakage</themekey>
                </theme>
        </keywords>
        <from_application application_ID="UGTL1">Underground Gas Tank Leakage (First
Approach)</from_application>
</idinfo>
<idinfo ID="BASEMAP">
        <name>Base map</name>
        <descript>A data set contains parcel and its attributes</descript>
        <keywords>
                <theme>
                        <themekey>underground gas</themekey>
                        <themekey>building</themekey>
                        <themekey>population</themekey>
                        <themekey>parcel</themekey>
                        <themekey>road</themekey>
                        <themekey>planning</themekey>
                </theme>
                <place>
                        <placekey>Canada</placekey>
                        <placekey>New Brunswick</placekey>
                </place>
        </keywords>
        <from_application application_ID="Digi">Manual Digitizing from hardcopy
maps</from_application>
</idinfo>
</spatialdata>
```

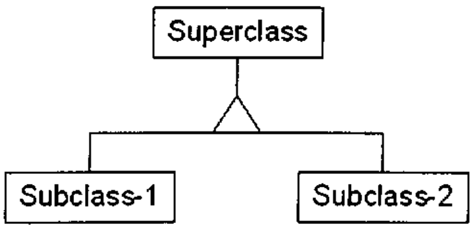
APPENDIX II – NOTATION FOR OBJECT MODELLING TECHNIQUE (OMT)

Object Model Notation

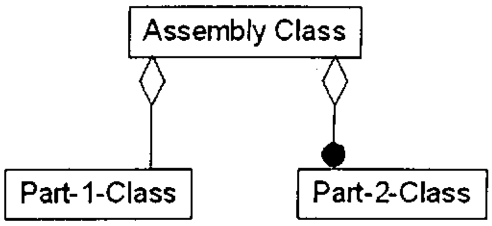
Class:


Class Name
Class Name
attribute
attribute : data_type
attribute : data_type = init_value
...
operation
operation (arg_list) : return_type
...

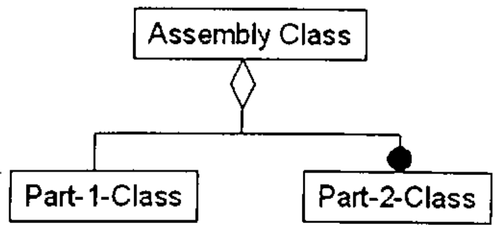
Generalization (Inheritance):



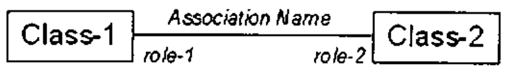
Aggregation:



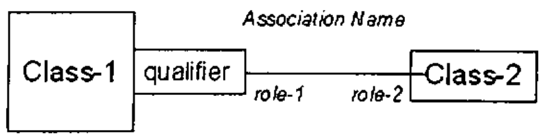
Aggregation (alternate form):



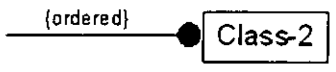
Association:



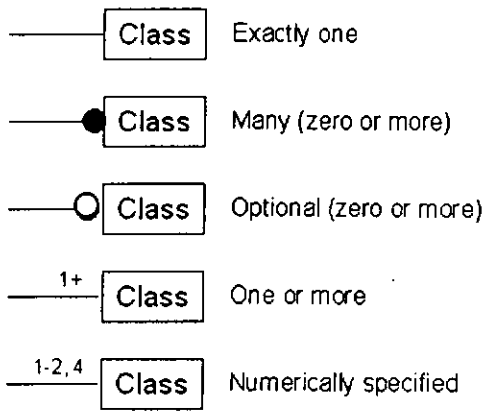
Qualified Association:



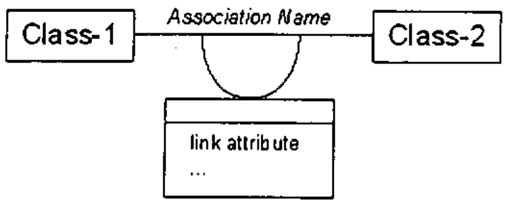
Ordering:



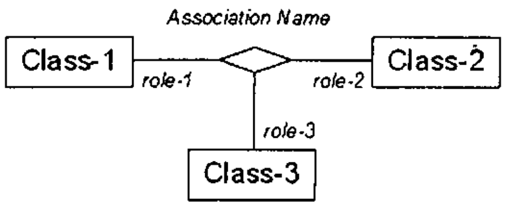
Multiplicity of Associations:



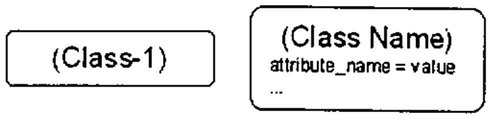
Link Attribute:



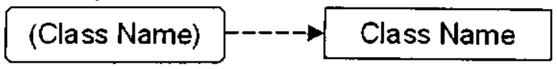
Ternary Association:



Object Instances:

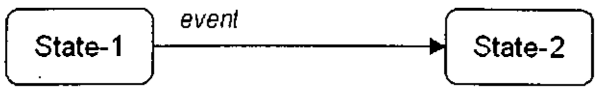


Instantiation Relationship:

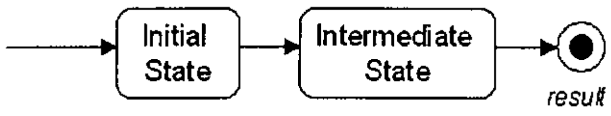


Dynamic Model Notation

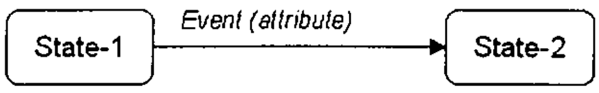
Event causes Transition between States:



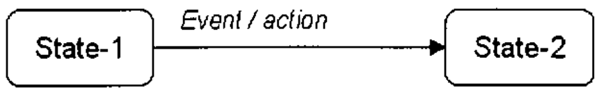
Initial and Final States:



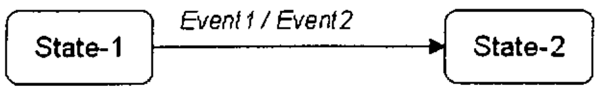
Event with Attribute:



Action on a Transition:

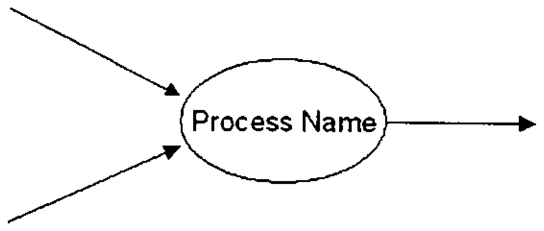


Output Event on a Transition:

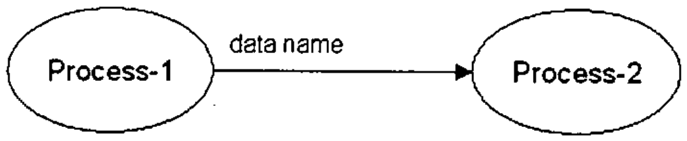


Functional Model Notation

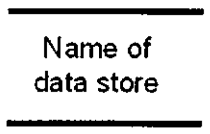
Process:



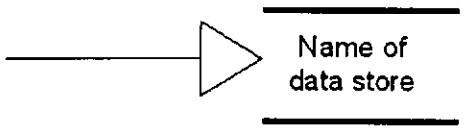
Data Flow between Processes:



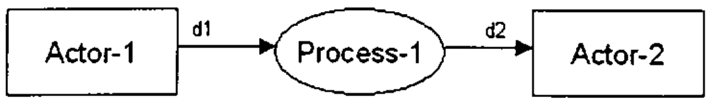
Data Store or File Object:



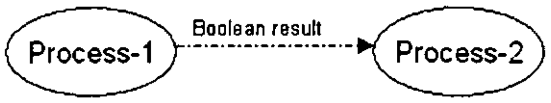
Data Flow that Results in a Data Store:



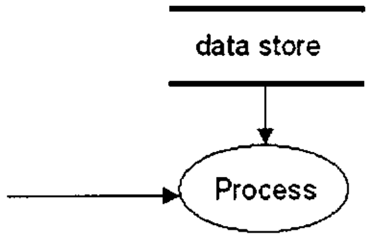
Actor Objects (as Source or Sink of Data):



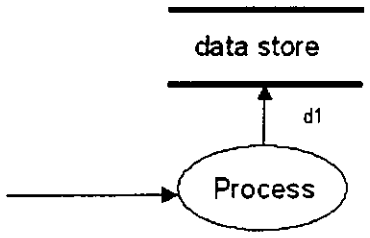
Control Flow



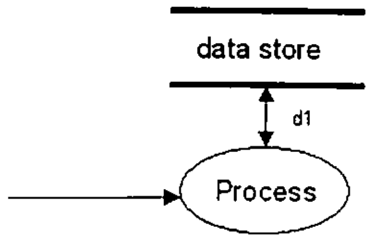
Access of Data Store Value:



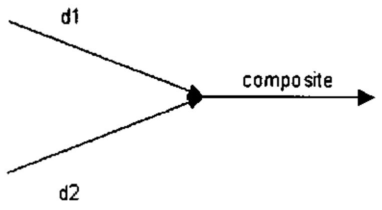
Update of Data Store Value:



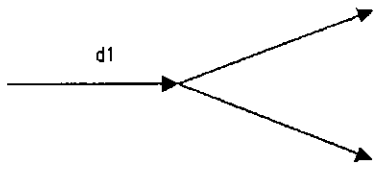
Access and Update of Data Store Value:



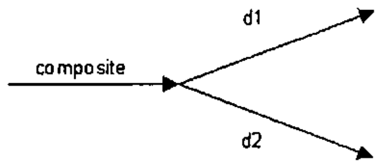
Composition of Data Value:



Duplication of Data Value



Decomposition of Data Value:



APPENDIX III – PETER SCHWEITZER'S ASCII TEMPLATE WITH TABBED INDENTS

```
Identification_Information:
      Citation:
            Citation_Information:
                  Originator:
                  Publication_Date:
                  Publication_Time:
                  Title:
                  Edition:
                  Geospatial_Data_Presentation_Form:
                  Series_Information:
                        Series_Name:
                        Issue_Identification:
                  Publication_Information:
                        Publication_Place:
                        Publisher:
                  Other_Citation_Details:
                  Online_Linkage:
                  Larger_Work_Citation:
                        Citation_Information:
      Description:
            Abstract:
            Purpose:
            Supplemental_Information:
      Time_Period_of_Content:
            Time_Period_Information:
                  Single_Date/Time:
                        Calendar_Date:
                        Time_of_Day:
                  Multiple_Dates/Times:
                        Calendar_Date:
                        Time_of_Day:
                  Range_of_Dates/Times:
                        Beginning_Date:
                        Beginning_Time:
                        Ending_Date:
                        Ending_Time:
            Currentness_Reference:
      Status:
            Progress:
            Maintenance_and_Update_Frequency:
      Spatial_Domain:
            Bounding_Coordinates:
                  West_Bounding_Coordinate:
                  East_Bounding_Coordinate:
                  North_Bounding_Coordinate:
                  South_Bounding_Coordinate:
            Data_Set_G-Polygon:
                  Data_Set_G-Polygon_Outer_G-Ring:
                        G-Ring_Latitude:
                        G-Ring_Longitude:
                  Data_Set_G-Polygon_Exclusion_G-Ring:
                        G-Ring_Latitude:
                        G-Ring_Longitude:
      Keywords:
            Theme:
                  Theme_Keyword_Thesaurus:
                  Theme_Keyword:
            Place:
                  Place_Keyword_Thesaurus:
                  Place_Keyword:
            Stratum:
                  Stratum_Keyword_Thesaurus:
                  Stratum_Keyword:
```

```
                Temporal:
                        Temporal_Keyword_Thesaurus:
                        Temporal_Keyword:
        Access_Constraints:
        Use_Constraints:
        Point_of_Contact:
                Contact_Information:
                        Contact_Person_Primary:
                                Contact_Person:
                                Contact_Organization:
                        Contact_Organization_Primary:
                                Contact_Organization:
                                Contact_Person:
                        Contact_Position:
                        Contact_Address:
                                Address_Type:
                                Address:
                                City:
                                State_or_Province:
                                Postal_Code:
                                Country:
                        Contact_Voice_Telephone:
                        Contact_TDD/TTY_Telephone:
                        Contact_Facsimile_Telephone:
                        Contact_Electronic_Mail_Address:
                        Hours_of_Service:
                        Contact_Instructions:
        Browse_Graphic:
                Browse_Graphic_File_Name:
                Browse_Graphic_File_Description:
                Browse_Graphic_File_Type:
        Data_Set_Credit:
        Security_Information:
                Security_Classification_System:
                Security_Classification:
                Security_Handling_Description:
        Native_Data_Set_Environment:
        Cross_Reference:
                Citation_Information:
                        Originator:
                        Publication_Date:
                        Publication_Time:
                        Title:
                        Edition:
                        Geospatial_Data_Presentation_Form:
                        Series_Information:
                                Series_Name:
                                Issue_Identification:
                        Publication_Information:
                                Publication_Place:
                                Publisher:
                        Other_Citation_Details:
                        Online_Linkage:
                        Larger_Work_Citation:
                                Citation_Information:
Data_Quality_Information:
        Attribute_Accuracy:
                Attribute_Accuracy_Report:
                Quantitative_Attribute_Accuracy_Assessment:
                        Attribute_Accuracy_Value:
                        Attribute_Accuracy_Explanation:
        Logical_Consistency_Report:
        Completeness_Report:
```

```
Positional_Accuracy:
      Horizontal_Positional_Accuracy:
            Horizontal_Positional_Accuracy_Report:

Quantitative_Horizontal_Positional_Accuracy_Assessment:
                  Horizontal_Positional_Accuracy_Value:
                  Horizontal_Positional_Accuracy_Explanation:
      Vertical_Positional_Accuracy:
            Vertical_Positional_Accuracy_Report:

Quantitative_Vertical_Positional_Accuracy_Assessment:
                  Vertical_Positional_Accuracy_Value:
                  Vertical_Positional_Accuracy_Explanation:
Lineage:
      Source_Information:
            Source_Citation:
                  Citation_Information:
                        Originator:
                        Publication_Date:
                        Publication_Time:
                        Title:
                        Edition:
                        Geospatial_Data_Presentation_Form:
                        Series_Information:
                              Series_Name:
                              Issue_Identification:
                        Publication_Information:
                              Publication_Place:
                              Publisher:
                        Other_Citation_Details:
                        Online_Linkage:
                        Larger_Work_Citation:
                              Citation_Information:
            Source_Scale_Denominator:
            Type_of_Source_Media:
            Source_Time_Period_of_Content:
                  Time_Period_Information:
                        Single_Date/Time:
                              Calendar_Date:
                              Time_of_Day:
                        Multiple_Dates/Times:
                              Calendar_Date:
                              Time_of_Day:
                        Range_of_Dates/Times:
                              Beginning_Date:
                              Beginning_Time:
                              Ending_Date:
                              Ending_Time:
                  Source_Currentness_Reference:
            Source_Citation_Abbreviation:
            Source_Contribution:
      Process_Step:
            Process_Description:
            Source_Used_Citation_Abbreviation:
            Process_Date:
            Process_Time:
            Source_Produced_Citation_Abbreviation:
            Process_Contact:
                  Contact_Information:
                        Contact_Person_Primary:
                              Contact_Person:
                              Contact_Organization:
                        Contact_Organization_Primary:
```

```
                                          Contact_Organization:
                                          Contact_Person:
                                     Contact_Position:
                                     Contact_Address:
                                          Address_Type:
                                          Address:
                                          City:
                                          State_or_Province:
                                          Postal_Code:
                                          Country:
                                     Contact_Voice_Telephone:
                                     Contact_TDD/TTY_Telephone:
                                     Contact_Facsimile_Telephone:
                                     Contact_Electronic_Mail_Address:
                                     Hours_of_Service:
                                     Contact_Instructions:
     Cloud_Cover:
Spatial_Data_Organization_Information:
     Indirect_Spatial_Reference:
     Direct_Spatial_Reference_Method:
     Point_and_Vector_Object_Information:
          SDTS_Terms_Description:
               SDTS_Point_and_Vector_Object_Type:
               Point_and_Vector_Object_Count:
          VPF_Terms_Description:
               VPF_Topology_Level:
               VPF_Point_and_Vector_Object_Type:
               Point_and_Vector_Object_Count:
     Raster_Object_Information:
          Raster_Object_Type:
          Row_Count:
          Column_Count:
          Vertical_Count:
Spatial_Reference_Information:
     Horizontal_Coordinate_System_Definition:
          Geographic:
               Latitude_Resolution:
               Longitude_Resolution:
               Geographic_Coordinate_Units:
          Planar:
               Map_Projection:
                    Map_Projection_Name:
                    Albers_Conical_Equal_Area:
                         Standard_Parallel:
                         Longitude_of_Central_Meridian:
                         Latitude_of_Projection_Origin:
                         False_Easting:
                         False_Northing:
                    Azimuthal_Equidistant:
                         Longitude_of_Central_Meridian:
                         Latitude_of_Projection_Origin:
                         False_Easting:
                         False_Northing:
                    Equidistant_Conic:
                         Standard_Parallel:
                         Longitude_of_Central_Meridian:
                         Latitude_of_Projection_Origin:
                         False_Easting:
                         False_Northing:
                    Equirectangular:
                         Standard_Parallel:
                         Longitude_of_Central_Meridian:
                         False_Easting:
```

```
                False_Northing:
            General_Vertical_Near-sided_Perspective:

Height_of_Perspective_Point_Above_Surface:
                Longitude_of_Projection_Center:
                Latitude_of_Projection_Center:
                False_Easting:
                False_Northing:
            Gnomonic:
                Longitude_of_Projection_Center:
                Latitude_of_Projection_Center:
                False_Easting:
                False_Northing:
            Lambert_Azimuthal_Equal_Area:
                Longitude_of_Projection_Center:
                Latitude_of_Projection_Center:
                False_Easting:
                False_Northing:
            Lambert_Conformal_Conic:
                Standard_Parallel:
                Longitude_of_Central_Meridian:
                Latitude_of_Projection_Origin:
                False_Easting:
                False_Northing:
            Mercator:
                Standard_Parallel:
                Scale_Factor_at_Equator:
                Longitude_of_Central_Meridian:
                False_Easting:
                False_Northing:
            Modified_Stereographic_for_Alaska:
                False_Easting:
                False_Northing:
            Miller_Cylindrical:
                Longitude_of_Central_Meridian:
                False_Easting:
                False_Northing:
            Oblique_Mercator:
                Scale_Factor_at_Center_Line:
                Oblique_Line_Azimuth:
                    Azimuthal_Angle:
                    Azimuth_Measure_Point_Longitude:
                Oblique_Line_Point:
                    Oblique_Line_Latitude:
                    Oblique_Line_Longitude:
                Latitude_of_Projection_Origin:
                False_Easting:
                False_Northing:
            Orthographic:
                Longitude_of_Projection_Center:
                Latitude_of_Projection_Center:
                False_Easting:
                False_Northing:
            Polar_Stereographic:
                Straight-Vertical_Longitude_from_Pole:
                Standard_Parallel:
                Scale_Factor_at_Projection_Origin:
                False_Easting:
                False_Northing:
            Polyconic:
                Longitude_of_Central_Meridian:
                Latitude_of_Projection_Origin:
                False_Easting:
```

```
                False_Northing:
          Robinson:
                Longitude_of_Projection_Center:
                False_Easting:
                False_Northing:
          Sinusoidal:
                Longitude_of_Central_Meridian:
                False_Easting:
                False_Northing:
          Space_Oblique_Mercator_(Landsat):
                Landsat_Number:
                Path_Number:
                False_Easting:
                False_Northing:
          Stereographic:
                Longitude_of_Projection_Center:
                Latitude_of_Projection_Center:
                False_Easting:
                False_Northing:
          Transverse_Mercator:
                Scale_Factor_at_Central_Meridian:
                Longitude_of_Central_Meridian:
                Latitude_of_Projection_Origin:
                False_Easting:
                False_Northing:
          van_der_Grinten:
                Longitude_of_Central_Meridian:
                False_Easting:
                False_Northing:
          Other_Projection's_Definition:
    Grid_Coordinate_System:
          Grid_Coordinate_System_Name:
          Universal_Transverse_Mercator:
                UTM_Zone_Number:
                Transverse_Mercator:
          Universal_Polar_Stereographic:
                UPS_Zone_Identifier:
                Polar_Stereographic:
          State_Plane_Coordinate_System:
                SPCS_Zone_Identifier:
                Lambert_Conformal_Conic:
                Transverse_Mercator:
                Oblique_Mercator:
                Polyconic:
          ARC_Coordinate_System:
                ARC_System_Zone_Identifier:
                Equirectangular:
                Azimuthal_Equidistant:
          Other_Grid_System's_Definition:
    Local_Planar:
          Local_Planar_Description:
          Local_Planar_Georeference_Information:
    Planar_Coordinate_Information:
          Planar_Coordinate_Encoding_Method:
          Coordinate_Representation:
                Abscissa_Resolution:
                Ordinate_Resolution:
          Distance_and_Bearing_Representation:
                Distance_Resolution:
                Bearing_Resolution:
                Bearing_Units:
                Bearing_Reference_Direction:
                Bearing_Reference_Meridian:
                        Planar_Distance_Units:
            Local:
                  Local_Description:
                  Local_Georeference_Information:
            Geodetic_Model:
                  Horizontal_Datum_Name:
                  Ellipsoid_Name:
                  Semi-major_Axis:
                  Denominator_of_Flattening_Ratio:
      Vertical_Coordinate_System_Definition:
            Altitude_System_Definition:
                  Altitude_Datum_Name:
                  Altitude_Resolution:
                  Altitude_Distance_Units:
                  Altitude_Encoding_Method:
            Depth_System_Definition:
                  Depth_Datum_Name:
                  Depth_Resolution:
                  Depth_Distance_Units:
                  Depth_Encoding_Method:
Entity_and_Attribute_Information:
      Detailed_Description:
            Entity_Type:
                  Entity_Type_Label:
                  Entity_Type_Definition:
                  Entity_Type_Definition_Source:
            Attribute:
                  Attribute_Label:
                  Attribute_Definition:
                  Attribute_Definition_Source:
                  Attribute_Domain_Values:
                        Enumerated_Domain:
                              Enumerated_Domain_Value:
                              Enumerated_Domain_Value_Definition:

      Enumerated_Domain_Value_Definition_Source:
            Attribute:
                  Attribute_Domain_Values:
                        Range_Domain:
                              Range_Domain_Minimum:
                              Range_Domain_Maximum:
            Attribute:
                  Attribute_Domain_Values:
                        Codeset_Domain:
                              Codeset_Name:
                              Codeset_Source:
                        Unrepresentable_Domain:
                  Attribute_Units_of_Measure:
                  Attribute_Measurement_Resolution:
                  Beginning_Date_of_Attribute_Values:
                  Ending_Date_of_Attribute_Values:
                  Attribute_Value_Accuracy_Information:
                        Attribute_Value_Accuracy:
                        Attribute_Value_Accuracy_Explanation:
                  Attribute_Measurement_Frequency:
      Overview_Description:
            Entity_and_Attribute_Overview:
            Entity_and_Attribute_Detail_Citation:
Distribution_Information:
      Distributor:
            Contact_Information:
                  Contact_Person_Primary:
                        Contact_Person:
```

```
                        Contact_Organization:
                Contact_Organization_Primary:
                        Contact_Organization:
                        Contact_Person:
                Contact_Position:
                Contact_Address:
                        Address_Type:
                        Address:
                        City:
                        State_or_Province:
                        Postal_Code:
                        Country:
                Contact_Voice_Telephone:
                Contact_TDD/TTY_Telephone:
                Contact_Facsimile_Telephone:
                Contact_Electronic_Mail_Address:
                Hours_of_Service:
                Contact_Instructions:
Resource_Description:
Distribution_Liability:
Standard_Order_Process:
        Non-digital_Form:
        Digital_Form:
                Digital_Transfer_Information:
                        Format_Name:
                        Format_Version_Number:
                        Format_Version_Date:
                        Format_Specification:
                        Format_Information_Content:
                        File_Decompression_Technique:
                        Transfer_Size:
                Digital_Transfer_Option:
                        Online_Option:
                                Computer_Contact_Information:
                                        Network_Address:
                                                Network_Resource_Name:
                                        Dialup_Instructions:
                                                Lowest_BPS:
                                                Highest_BPS:
                                                Number_DataBits:
                                                Number_StopBits:
                                                Parity:
                                                Compression_Support:
                                                Dialup_Telephone:
                                                Dialup_File_Name:
                                Access_Instructions:
                                Online_Computer_and_Operating_System:
                        Offline_Option:
                                Offline_Media:
                                Recording_Capacity:
                                        Recording_Density:
                                        Recording_Density_Units:
                                Recording_Format:
                                Compatibility_Information:
        Fees:
        Ordering_Instructions:
        Turnaround:
Custom_Order_Process:
Technical_Prerequisites:
Available_Time_Period:
        Time_Period_Information:
                Single_Date/Time:
                        Calendar_Date:
```

```
                    Time_of_Day:
                Multiple_Dates/Times:
                    Calendar_Date:
                    Time_of_Day:
                Range_of_Dates/Times:
                    Beginning_Date:
                    Beginning_Time:
                    Ending_Date:
                    Ending_Time:
Metadata_Reference_Information:
    Metadata_Date:
    Metadata_Review_Date:
    Metadata_Future_Review_Date:
    Metadata_Contact:
        Contact_Information:
            Contact_Person_Primary:
                Contact_Person:
                Contact_Organization:
            Contact_Organization_Primary:
                Contact_Organization:
                Contact_Person:
            Contact_Position:
            Contact_Address:
                Address_Type:
                Address:
                City:
                State_or_Province:
                Postal_Code:
                Country:
            Contact_Voice_Telephone:
            Contact_TDD/TTY_Telephone:
            Contact_Facsimile_Telephone:
            Contact_Electronic_Mail_Address:
            Hours_of_Service:
            Contact_Instructions:
    Metadata_Standard_Name:
    Metadata_Standard_Version:
    Metadata_Time_Convention:
    Metadata_Access_Constraints:
    Metadata_Use_Constraints:
    Metadata_Security_Information:
        Metadata_Security_Classification_System:
        Metadata_Security_Classification:
        Metadata_Security_Handling_Description:
```

VITA

CHAN HO CHIU, ELTON
Department of Land Surveying & Geo-Informatics,
The Hong Kong Polytechnic University
Kowloon, Hong Kong, China
9798
(852) 2766-

Personal

Born	Hong Kong
Citizenship	Hong Kong, China

Education

Master of Philosophy in Geographic Information Systems 1999

The Hong Kong Polytechnic University, Hong Kong, China

Thesis: "Design of Application Model for Geospatial Data Clearinghouse"

Supervisor: Dr. Lee Yuk-cheung

Bachelor of Science (2nd Honor, Class A) in Land Surveying & Geo-Informatics 1994

The Hong Kong Polytechnic University, Hong Kong, China

This page is intentionally left blank.