

DEVELOPMENT OF A GEOSPATIAL REFERENCE FRAMEWORK – A CASE STUDY FOR THE UNB-GGE SURVEY CAMP

SAMYAR SEPEHR

December 2013



**TECHNICAL REPORT
NO. 288**

**DEVELOPMENT OF A GEOSPATIAL
REFERENCE FRAMEWORK – A CASE
STUDY FOR THE UNB-GGE SURVEY
CAMP**

Samyar Sepehr

Department of Geodesy and Geomatics Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada
E3B 5A3

December 2013

© Samyar Sepehr, 2013

PREFACE

This technical report is a reproduction of a thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Department of Geodesy and Geomatics Engineering, December 2013. The research was supervised by Dr. Emmanuel Stefanakis.

As with any copyrighted material, permission to reprint or quote extensively from this report must be received from the author. The citation to this work should appear as follows:

Sepehr, Samyar (2013). *Development of a Geospatial Reference Framework – A Case Study for the UNB-GGE Survey Camp*. M.Sc.E. thesis, Department of Geodesy and Geomatics Engineering, Technical Report No. 288, University of New Brunswick, Fredericton, New Brunswick, Canada, 156 pp.

ABSTRACT

This thesis describes the development of a geospatial reference framework for categorizing, organizing, validating, browsing, and representing survey camp topographic data. This topographic data is collected annually by the Geodesy and Geomatics Engineering (GGE) students at the University of New Brunswick (UNB) as part of the requirements for a UNB course. ESRI ArcGIS 10 was used to build the information products associated with the geospatial framework. The information products were employed for analyzing, organizing, and managing the past and future topographic map collections. In order to make the geospatial reference framework easily accessible, a Web-GIS application was developed using ArcGIS Server on the server side and ArcGIS JavaScript API on the client side.

This thesis represents the establishment of an appropriate geodatabase model that has been designed and built to satisfy the requirements and characteristics defined by the spatial reference framework. The project contributed in designing and producing geospatial information products including a geographical repository of UNB campus, geospatial data validation tools, repository maintenance methods, and the Web-GIS service. These geospatial information products constitute a geospatial reference framework. This framework allows for organization, storage, and representation of past survey camp data collections and provides the specifications and standards for the future collections.

Acknowledgment

I gratefully acknowledge the following people for assessing and encouraging me to complete this research project.

- Dr. Emmanuel Stefanakis, my supervisor, for his support, guidance, and encouragement. I am very thankful for the time and patience he took to give me the valuable comments and feedbacks.
- Dr. Susan E. Nichols, for her helpful advices and guidance towards my Master's program.
- Mr. David Fraser for answering countless questions and giving great advice.
- Sylvia Whitaker who has always assisted me with all administrative issues.
- My dear family, my parents, for their endless support and encouragement.
- My dear friends for their assistance and encouragement.

Table of Contents

Abstract.....	ii
Acknowledgment	iii
Table of Contents.....	iv
List of Tables.....	vii
List of Figures.....	viii
Chapter 1 Introduction.....	1
1.1 Thesis Background.....	2
1.2 Thesis Agenda and Objectives.....	5
1.2.1 Objective.....	6
1.3 Contribution.....	6
1.4 Thesis Outline.....	7
Chapter 2 System and Methodology.....	8
2.1 Introduction.....	8
2.2 Background.....	9
2.3 Development Workflow.....	10
2.3.1 Data Assessment.....	13
2.3.2 Framework Characteristics.....	14
2.3.3 Building Geodatabase.....	15
2.3.3.1 ESRI Local Government Geodatabase Model.....	17
2.3.4 Data Validation and Geodatabase Maintenance.....	21
2.3.5 Web-GIS Service.....	22
2.4 Summary.....	24
Chapter 3 Survey Camp Data Assessment and Framework Characteristics 25	
3.1 Introduction.....	25
3.2 Survey Camp Data Assessment.....	26
3.2.1 Lack of Consistency in Features Coding System.....	27
3.2.2 Inappropriate Feature Encoding.....	29
3.2.3 Incorrect Feature Attributes Coding.....	31
3.2.4 Undefined Shared Geometries.....	32
3.2.5 Unenclosed Areal Features.....	33
3.2.6 Inconsistent Survey of Treed Areas and Green Fields.....	34
3.2.7 Inconsistent spatial location of features.....	36
3.3 Framework Characteristics.....	37
3.3.1 Consistence Feature Surveying.....	38
3.3.2 Feature Coding Schema.....	39

3.3.3	Spatial Representation Forms.....	44
3.3.4	Shared Feature Types.....	45
3.3.5	Spatial Reference System.....	46
3.4	Summary.....	46
Chapter 4	Geodatabase.....	47
4.1	Introduction.....	47
4.2	Geodatabase Design Phases.....	48
4.3	Conceptual Design.....	50
4.3.1	Information Products.....	50
4.3.2	Thematic Layers Specifications.....	51
4.3.3	Datasets Specifications.....	53
4.4	Logical Design.....	55
4.4.1	Define the Tabular Structure and Behavior.....	55
4.4.2	Define the Spatial Properties of the Datasets.....	57
4.4.3	Propose a Geodatabase Design.....	58
4.5	Physical Design.....	59
4.5.1	Implement and Prototype Geodatabase Design.....	59
4.5.1.1	Creating Geodatabase and Loading Data.....	61
4.5.1.2	Building the Topologies and Testing the Model.....	63
4.5.2	Design workflows for Maintenance.....	67
4.5.2.1	GIS models for Geodatabase Maintenance.....	71
4.5.3	Design Documentation.....	75
4.6	Lessons Learned.....	77
Chapter 5	Web-GIS Service Development.....	78
5.1	Introduction.....	78
5.2	Web-GIS System.....	79
5.2.1	GIS Server.....	81
5.2.2	Client Side.....	83
5.3	Web-GIS Programming Structure and Properties.....	84
5.3.1	Input Data and Data Type.....	86
5.3.2	Functionalities.....	88
5.3.3	Calling the Functions.....	92
5.3.4	Output Data and Data Type.....	94
5.4	Examining the Map Service Performance.....	96
5.5	Summary.....	99
Chapter 6	Conclusions.....	100
6.1	Research Outcomes and Issues Encountered.....	101

6.2	Recommendations for Future Research.....	104
Bibliography.....		106
Appendix I Geodatabase Maintenance.....		109
I.1	Validating the Features' Codes.....	109
I.2	Validating the Features' Data Type.....	112
I.3	Applying GIS models.....	113
I.4	Glossary of GIS Terms.....	118
Appendix II Geodatabase Schema Documentation.....		121
Appendix III Web Service User Guide and Code.....		126
III.1	User Guide.....	126
III.2	ArcCatalog Manager.....	129
III.3	HTML and JavaScript Codes.....	130

Vita

List of Tables

Table 2.1: Comparison of the Local Government feature classes with the survey camp's data.....	18
Table 2.2: Specifications and advantages of the survey camp geodatabase.....	20
Table 3.1: Summary of the students' feature codes of the map sets of years 2012 and 2011.....	41
Table 3.2: Summary of the campus features attribute codes.....	43
Table 3.3: Spatial representation forms of the campus features.....	44
Table 3.4: Examples of the features with shared geometry.....	45
Table 4.1: Phases and steps of designing a geodatabase (Arctur and Zeiler, 2004).....	49
Table 4.2: Specifications of the campus key thematic layers.....	51
Table 4.3: Specifications of the campus key thematic layers.....	52
Table 4.4: Datasets and feature classes.....	54
Table 4.5: Geodatabase coded value domains.....	56
Table 4.6: Topology rules.....	58
Table 5.1: ArcGIS JavaScript API classes and properties utilized for input data.....	88
Table 5.2: List of the ArcGIS JavaScript API classes for the visibility controller.....	89
Table 5.3: The JavaScript classes utilized in the QueryTask execution function.....	91
Table 5.4: List of the layers and attributes published by the map service.....	95

List of Figures

Figure 2.1: Methodology workflow for the spatial framework development.....	12
Figure 3.1: Example of a building feature coded as “B”	27
Figure 3.2: Example of a building feature with no specific code	28
Figure 3.3: Example of the features encoded with polylines.....	30
Figure 3.4: Example of the center lines coded incorrectly as buildings	31
Figure 3.5: Example of incorrect coding for the shared geometries between the buildings and parking lots	32
Figure 3.6: Comparison of the parking lot polygons with the CAD polylines.....	33
Figure 3.7: Example of the trees’ points and the enclosed lines of the wooded area	35
Figure 3.8: Map sets of the survey camp of years 2012, 2011	40
Figure 4.1: Implementation work flow for building and populating the geodatabase	60
Figure 4.2: Example of tree points specified by query expression.....	61
Figure 4.3: Example of parking lots and green fields before the editing.....	62
Figure 4.4: Example of parking lots and green fields after the editing	63
Figure 4.5: Reports of the topological errors of each dataset	64
Figure 4.6: Structure of the geodatabase created in the ArcCatalog	65
Figure 4.7: 1:6000 survey camp base map.....	66
Figure 4.8: Example of 1:2500 survey camp base map.....	67
Figure 4.9: Geodatabase maintenance workfellow.....	71
Figure 4.10: Shows the tools and processes performed in the GIS models.....	74
Figure 4.11: Example of dataset documentation.....	76

Figure 5.1: General illustration of the Web-GIS system and architecture.....	80
Figure 5.2: ArcGIS server system architecture.....	82
Figure 5.3: Shows the programming framework and libraries.....	83
Figure 5.4: General aspects of the Web-GIS service code and performance.....	85
Figure 5.5: Visibility controller function of the map service.....	89
Figure 5.6: Workfellow of the QueryTask execution function.....	91
Figure 5.7: Layers' IDs in the REST service API.....	93
Figure 5.8: Example of the map service with no selected features	96
Figure 5.9: Example of representing features selected in the checklist	97
Figure 5.10: Example of querying the features on the map service	98
Figure I.1: Summarization of the attributes codes in the Layer field.....	109
Figure I.2: Domain table.....	110
Figure I.3: Example of the invalid codes.....	111
Figure I.4: Example of the wrong feature coding.....	112
Figure I.5: Example of the validated attributes.....	113
Figure I.6: Example of the GIS model creating feature classes.....	114
Figure I.7: Example of old and new building polygons.....	115
Figure I.8: Example of the GIS model performs the updating process.....	116
Figure I.9: Examples of the results for the geoprocessing steps of the GIS model.....	117
Figure II.1: Overview of UNB campus geodatabase structure.....	121
Figure II.2: Route dataset.....	122
Figure II.3: Building-Wall-Contoure-Parking Dataset.....	123

Figure II.4: GreenArea-Infrastructure Dataset.....	124
Figure II.5: Domain table.....	125
Figure III.1: Example of map service representing the background base map.....	126
Figure III.2: Example of selected layers to display on the map service.....	127
Figure III.3: Example of pop-up window indicating features attribute data.....	128
Figure III.4: Example of the map service preview in ArcCatalog.....	129

Chapter 1 Introduction

Geographical information system (GIS) as a combination of information and geospatial technologies is a tool that results in better representing, organizing, and managing of spatial and non-spatial data. GIS allows improving the progress of information accessing in terms of providing the tools for analysing, processing, and then producing informative spatial data with reliable quality and consistency. The purpose of a GIS is to provide a spatial framework to support decisions for the intelligent use of earth's resources and to manage the man-made environment (Zeiler, 1999). GIS is used for integrated management of spatial and attribute data for various types of facilities and to provide users with information models illustrating the data structure with regard to the spatial and tabular behaviors.

Many organizations that manage geospatial information use GIS technologies to develop information models of their holdings. According to the previous researches and projects regarding campus GIS, universities are organizations which significantly demanding information technologies such as GIS to build management and representation systems for their geospatial holdings. Different applications built in order to organize and represent the campus geospatial data such as campus Web-GIS (Fangli et al, 2010), campus information navigation system based on GIS (Huang et al, 2010), and campus spatial information service (Yang, 2009).

In this thesis, a set of geospatial information products were developed according to the requirements of the GIS group in Geodesy and Geomatics Engineering (GGE) department of University of New Brunswick (UNB) to manage students surveyed topographic data. This data was collected as part of the requirements for GGE 2013 - course often referred to as “Survey Camp II”. The applications and the information products built in this thesis can be viewed as a geospatial reference framework for this GGE survey camp. This is because the framework provides students with surveying specifications that they need to consider (i.e. the topographic features that need to survey, the procedures that need to follow when collecting and then creating the map products, etc.) and various tools, methods and services for validating the student surveyed work outcomes. Thus, the overall goal of developing the geospatial reference framework is to improve and facilitate the processes of analyzing, validating, categorizing, storing, retrieving and representing GGE survey camp geospatial data. The development of the geospatial reference framework will be explained in terms of the information systems that have been produced including the geodatabase model, validation and maintenance processes, and the Web-GIS service.

1.1 Thesis Background

According to Delliska et al, 2008, GIS and information models or products that are utilized for managing geospatial data of the universities’ campus commonly provide and perform the followings items:

- Analysing the existing maps and plans of university facilities and features;
- Visualizing of maps and plans of university regions and features such as buildings, road networks, parks, sport centers, infrastructure objects etc.; and
- Identifying and generating of campus key thematic layers.

In this thesis, the information model comprised the core of the campus GIS system. The information model is an ESRI geodatabase. A geodatabase (GIS or -geographical database) has been designed and implemented focusing on the GGE survey camp geospatial holdings, with respect to the standard procedures of building geodatabases; including conceptual, logical, and physical design phases (Arctur and Zeiler, 2004). Designing a geodatabase involves defining the thematic layers in terms of the usage, content, and representation of each thematic layer or defining how geographic features are to be represented (for example, as points, lines, polygons, or tabular attributes). Furthermore, geodatabase design is to specify how data is organized into feature classes and attributes and also specifying the GIS behaviours by establishing the spatial relationships in datasets using topology rules (Arctur and Zeiler, 2004). After accomplishing the design and building phases, the geodatabase was populated with the survey camp geospatial data originally obtained from the CAD files of the UNB campus topographic maps as surveyed and created by the GGE undergraduate students.

Prior to designing the geodatabase, the original CAD files were validated and analyzed using ArcGIS 10 in order to address the issues and difficulties in terms of the data consistency and variety. The geospatial reference framework requirements were specified with respect to the results obtained from CAD files assessment in order to solve, manage, and minimize the issues and difficulties. Furthermore, the validation methods were introduced and defined with respect to the requirements first to improve the data quality and consistency and then to support and facilitate the geodatabase maintenance processes and workflow.

The maintenance workflow of the geodatabase consists of specifying, converting, validating, editing, and finally updating the data in the repository. First, the CAD data (map layers) must be converted to GIS data and then analyzed and validated based on the features' category, representation form, and the attribute coding system defined as the information model conceptual and logical specifications. Moreover, a Web-GIS service was developed to publish an interactive base-map which consists of the feature layers and associating attribute codes of UNB campus to help the users understand the geodatabase structure in terms of the features' spatial and non-spatial properties. The Web-GIS service built using ArcGIS server and ArcGIS JavaScript API to visualize the campus' feature layers along with performing a set of identification and query functions.

1.2 Thesis Agenda and Objectives

The purpose of this Master's Thesis is to develop and propose a geospatial reference framework to manage and organize the student surveyed topographic data. The geospatial framework consists of a set of specifications and information products which improve and facilitate the processes of categorizing, arranging, storing, identifying, and representing the survey camp data including:

Specifications:

- Defining the key feature layers (thematic layers) of the campus.
- Establishing a feature coding schema for the key features.
- Specifying the spatial data types (geometry types) of the key features.
- Defining the spatial relationships between the features.

Information Products:

- A geodatabase model to store and then to control the survey camp data spatial and non-spatial behaviours according to the specifications mentioned above.
- A Web-GIS application to allow the users to visually explore and interpret the campus' features layers and also to query the tabular coded value, and therefore facilitate the process of representing the framework specifications.

1.2.1 Objectives

1. To define the spatial and non-spatial characteristics of the geospatial reference framework; this will allow for establishing the validation methods as well as defining specifications to be used during the survey camp operation.
2. To create surveying specifications (e.g. what features to survey, which feature codes to assign to feature) to be used by students during topographic survey. These specifications will improve data consistency and facilitate repository maintenance process.
3. To design and build a geographical repository, a geodatabase, in order to categorize, store, and manage the GGE survey camp geospatial data.
4. To develop a Web-GIS service to publish an interactive base map of the UNB campus in order to represent the spatial framework characteristics and specifications.

1.3 Contributions

The overall goal of this project's objectives is to create a GIS system to analyse and manage GGE survey camp data. The GIS approaches and solutions, information products, and applications built in this project will be employed in the training purposes. Respectively, students will have a different and new interpretation of the survey camp geospatial data. Moreover, by using the GIS applications and information products, the survey data quality and consistency will be improved.

1.4 Thesis Outline

This thesis is organized into six chapters. This chapter introduces the thesis backgrounds, objectives, and contributions. Chapter 2 discusses the methodology carried out to achieve the objectives as well as reviewing the technologies and terms employed to produce the project's major outcomes. Chapter 3 includes the data examination and validation procedure as well as defining the systems and methods of establishing the geospatial reference framework's characteristics. Chapter 4 explains the process of designing and implementing the geodatabase along with illustrating the maintenance workflow and documentation of the database. Chapter 5 discusses the Web-GIS service development procedure along with showing the results and highlights the advantages of the service. Chapter 6 discusses the thesis outcomes and issues encountered. Recommendations for the future research are provided as well.

Chapter 2 System and Methodology

2.1 Introduction

In this chapter a background of the survey camp will be given to illustrate its outcomes and regulations as well as the issues and inconsistencies occurred in surveying camp data, due to the differences in specifications, usage, and standards of CAD and GIS systems. The methodology of the spatial reference framework development will be introduced and illustrated. The methodology procedure consists of three phases and each phase will be discussed in terms of the applying technology as well as the information systems and services produced.

In the first phase, the importance of survey camp data validation will be discussed in order to specify the causes of the issues existed in the CAD files. In the second and the third phase, the information system and Web-GIS service developed in this project will be discussed to illustrate their usage, advantages, and the capabilities in managing and organizing the survey camp data. Furthermore, the information products that were utilized in other research work, have been described briefly to illustrate the methods applied for organizing and managing various types of spatial data. Besides, the spatial data management methods, information products, and GIS approaches that were applied in others' research work have been reviewed. Also, a comparison performed to evaluate the suitability of a geodatabase model which is similar to the one that was developed in this thesis.

2.2 Background

Each year, GGE students conduct a topographic survey of part of the UNB campus as part of the course requirements for GGE2013, this course is often referred to as Survey Camp II. Students work in groups and each group is assigned a specific area of UNB campus. Each survey area typically contains several types of features such as buildings, sidewalks, streets, parking lots, green areas, trees, lamp posts, etc. Each group is responsible for creating a digital CAD file and a topographic map of their survey area and these two products are typically created using CAD software. In comparison with Orthophotos (acquired in 2008, resolution is approx 15 cm), the student's survey data generally indicate the following specifications:

- The collected data tends to correctly depict the location of features on the UNB campus.
- The data is complete, that is, contains the topographic features that students were supposed to survey.

However, after examining the students' surveyed data, there are many issues with the digital datasets that need to be resolved before incorporating the data into a campus GIS. The issues will be discussed in detail in chapter 3. Most of the issues occurred due to the requirements and specifications of the GIS system, which were not met in the students' maps generated in CAD environment. These specifications and requirements mostly influence the data identification, categorization, validation, and storage processes in the GIS system.

In a GIS system, the spatial and non-spatial properties of the data must be defined to improve data storage, visualization and retrieval. Therefore, to build a GIS for the GGE survey camp, first the existing issues of the students' CAD files were analyzed and outlined and then the spatial framework was developed according to the specifications and requirements defined against the existing issues.

2.3 Development Workflow

As mentioned, the methodology consists of three phases; each phase includes the following steps:

- ***Phase one:***

- 1- To assess and examine the CAD files of survey camp topographic maps.
- 2- To specify the spatial reference framework's characteristics.

- ***Phase two:***

- 3- To design and implement the GIS repository (geodatabase).
- 4- To introduce the methods and workflow of validating the data and maintaining the repository.

- ***Phase three:***

- 5- To develop the Web-GIS service.

The first phase will be briefly described in this chapter to outline the procedures and purposes of the data validation. Later in the chapter 3, the whole assessment process will be explained along with the examples to highlight the issues. The second and the third phases consist of the development and implementation of the main information products including the geodatabase, data validation methods, geodatabase maintenance workflow, and the Web-GIS service.

The details of the information product development will be explained separately in the later chapters. Figure 2.1 shows a flowchart including the developing approaches and processes implemented in the spatial framework development procedure.

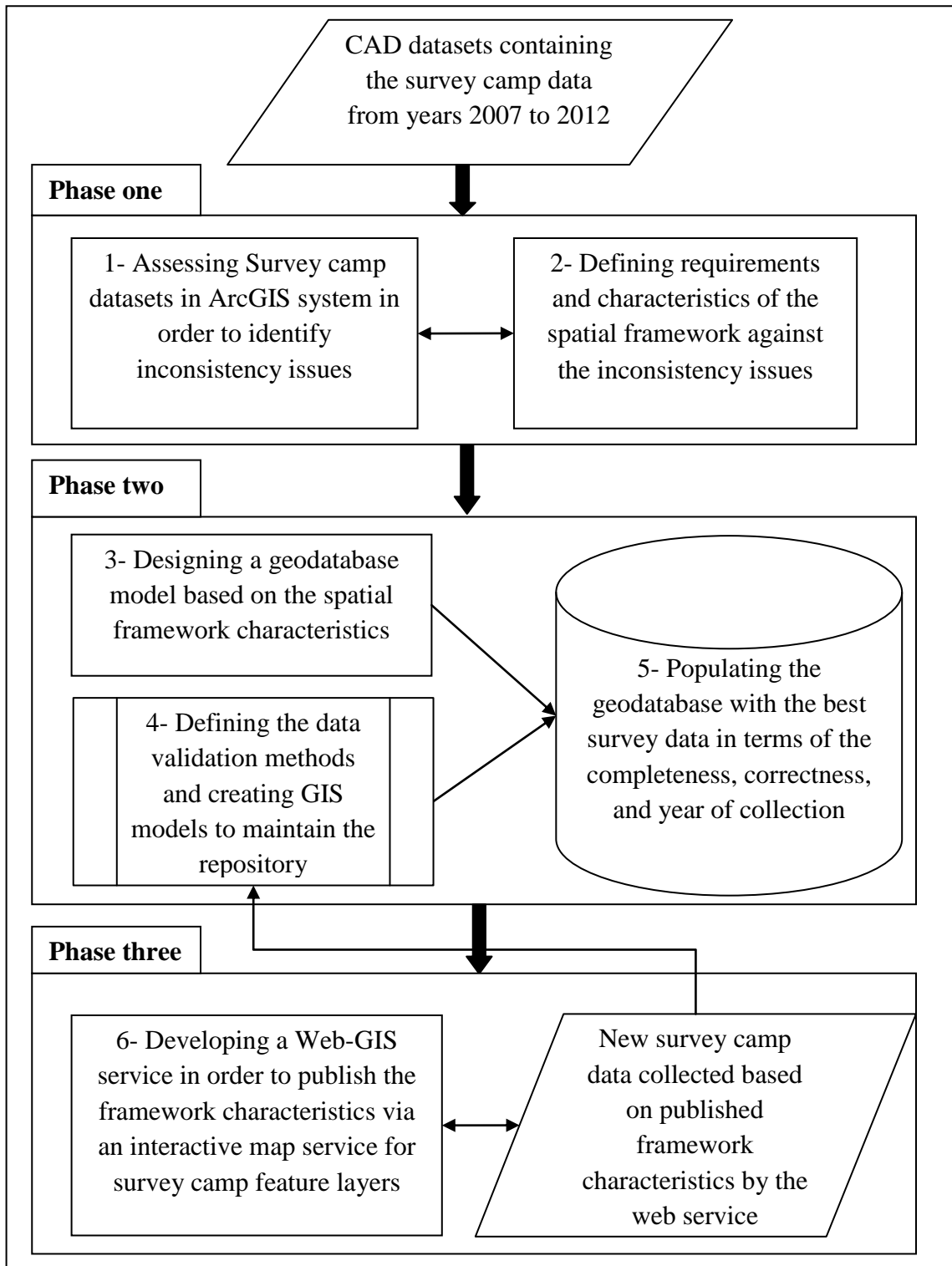


Figure 2.1: Methodology workflow for the spatial framework development

2.3.1 Data Assessment

By implementing the data assessment, the problems and issues with the data were identified. The survey camp data issues caused the major difficulties in building the GIS system. Moreover, identifying the issues was necessary prior to developing a spatial framework which could facilitate the survey camp data management; specifically, identifying the issues helped in establishment of the framework specifications. The steps mentioned below were performed in order to assess the data:

1. Creating an initial geodatabase and then establishing the datasets according to the years of survey data collection.
2. Importing the CAD files as feature classes into the corresponding datasets according to the year of collection.
3. Adding the data into ArcMap and then analyzing the attributes of the “*Layer*” field and the spatial data types of each feature. For example, building features were analysed in various years and the issue identified regarding the attribute codes associated with the features. Additionally, data type of the features in the CAD maps could not meet the logical requirements of the GIS system. For instance, features must be stored and represented as polygons, lines, and points; however, features in the CAD files were mostly represented with polylines. The issues identified in the CAD files cause the identification and classification of survey camp data to be hard and time consuming. The geospatial reference framework specifications were introduced considering the issues. The information products were developed to implement the specifications.

2.3.2 Framework Characteristics

The framework characteristics were defined in a way to support the information products to fulfill the goal of organizing the survey camp data. The following items were supported in the framework characteristics:

1. Specifying the classes and layers of the features which must be surveyed and then represented in the CAD maps. Therefore, the surveying groups will know that what features are expected to be surveyed. Accordingly, the CAD files that containing the new survey data, would be appropriate in terms of data consistency and particularly ready to be stored and used in the GIS repository.
2. Providing an attribute coding schema in order to improve and facilitate the process of identifying, validating and then organizing the survey camp data. This item specifically allowed the GIS system to control the tabular behavior of the geospatial data.
3. Specifying the data spatial type for the various features existing on the campus. This item significantly improved the mapping and cartography purposes. Additionally, the new survey camp data will be mapped according to the spatial data type that defined by the framework.
4. Establishing the topology rules to control the spatial consistency of the key features.

The characteristics mentioned above will be explained with the examples in section 3.5. The characteristics were specifically considered in designing and building the geodatabase, validation and maintenance methods, and developing the Web-GIS service.

2.3.3 Building the Geodatabase

Relational Database Management System such as the ESRI geodatabase has been used extensively in the last decade for the compilation of data as well as ArcGIS for visualizing the spatial data (Chesnaux .R et al, 2011). There are various research projects regarding establishing geodatabase models for different types of spatial data. Some of those researches will be mentioned in this section first to demonstrate the usage of geodatabase and then to compare other projects with this research project. Furthermore, a geodatabase model that generated by ESRI to manage campus facilities will be addressed and then analysed in ArcCatalog to show the similarities and differences with the geodatabase model created in this project for organizing the survey camp data.

- ***Mapping hydrogeological features:*** In this research project a spatial database implemented utilizing ESRI personal geodatabase as a repository of hydrological spatial data (Chesnaux .R et al, 2011). The geographical information organized into datasets created from Microsoft Access which is a relational database. Various spatial data such as, topography, wetlands, land use, administrative, transportation, and hydrogeological content were added and organized as feature classes in the geodatabase.
- ***Storm-water management:*** A campus GIS developed in this project to provide a spatial and tabular framework for storm-water system management on North Carolina State University (NCSU) (Smith and Devine, 2006). Moreover, the GIS system is an evaluation tool to measure the performance in the storm-water management plan. The data layers stored in the geodatabase datasets as polygons,

polylines, and points to generate the NCSU campus basemap. Storm-water data developed utilizing editing session in ArcGIS and then stored in the geodatabase.

- ***Urban information system:*** A geodatabase designed for urban thematic layers of Hanoi city including basemap layer, administrative area, cadastre layer, inhabitant area, transport layer, Environment layer, society – economic layer, and infrastructure layer (Binh et al, 2008). The entities were stored and organized as feature classes in the geodatabase. For example land parcels and houses specified with polygons or streets and trees specified as lines and points respectively. Moreover, the topology rules established for the feature classes to maintain the data integrity and consistency in the geodatabase.
- ***Urban road network environmental quality:*** The research placed in the city of Chania (GR) and the goal was to develop a geodatabase model in order to evaluate the environmental quality of the urban roads (Tsouchlaraki A. et al, 2010). Geodatabase design consisted of four stages as: defining the geodatabase, conceptual design, logical design, and physical design of the geodatabase. Eight main categories defined to address the tabular behaviour of the road features as urban planning and architectural, construction materials, road equipments, road traffic, land uses, pollution, climatic, and economic indices. The street lines generated by digitizing the axis of road network in the study area.
- ***Web-GIS for Wetlands:*** The objective of this project was to develop a web-GIS and a geodatabase for Florida's wetland providing map and data services (Mathiyalagan V. et al, 2005). The geodatabase utilized as a repository to

respond the queries implementing in the Web-GIS regarding the wetland's data such as geographic location, map projection, vegetation type, and soil property.

2.3.3.1 ESRI Local Government Geodatabase Model

The Local Government geodatabase model supports the following:

1. Local government basemap
2. Infrastructure maps and apps
3. Election maps and apps
4. Public safety maps and apps
5. Planning and developing maps and apps
6. Facilities and campus maps and apps
7. Address maps and apps
8. Public works maps and apps

(<http://www.arcgis.com/home/item.html?id=b8905e21104342afbe830da28d11b2b9>)

The model was downloaded from the URL mentioned above in order to analyse the geodatabase structure in terms of its datasets and feature classes. The Local Government geodatabase was opened in ArcCatalog and then the datasets and feature classes were compared with the survey camp data. The “FacilitiesStreets”, “SewerStormwater”, and “waterDestribution” datasets were determined to most closely resemble the contents in the survey camp CAD files. Table 2.1 shows examples of how the feature classes in these datasets compared with the survey camp data content.

Table 2.1: Comparison of the Local Government feature classes with the survey camp's data.

Feature classes of FacilitiesStreets Dataset	If supports the survey camp data content	Matching data in the survey camp
BridgePoint	No	-
Building	Yes	Building polylines
BuildingFloor	No	-
BuildingFloorPlanLine	No	-
BuildingFloorSection	No	-
BuildingInteriorSpace	No	-
BuildingPhotoLocation	No	-
CurbRamp	Yes	Curb polylines
Gurdrail	Yes	Fence polylines
LandscapeArea	Yes	Green areas
ParkingSpace	Yes	Parking areas polylines
PavementMarkingLine	No	-
PavementMarkingPoint	No	-
PavementSchedule	No	-
Pole	No	-
RRCrossing	No	-
Sidewalk	Yes	Sidewalk polylines
SiteAmenityLine	No	-
Street	Yes	Street polylines
StreetFurniture	No	-
StreetIntersection	No	-
StreetPavement	Yes	Asphalt areas polylines
Tree	Yes	Tree points
SewerStormwater Dataset		
Stormwater_Net_Junction	No	-
ssCleanOut	No	-
ssDetention	No	-
ssFitting	No	-
ssManhole	Yes	Manhole points
ssOpendrain	Yes	Drain storm points
ssSystemValve	No	-
ssTap	No	-
ssTestStation	No	-
WaterDistribution Dataset		
wHydrant	Yes	Hydrant points
wPump	No	-
wSystemValve	No	-
wTestStation	No	-

According to the comparison illustrated above, there are three reasons to not use the Local Government geodatabase as an information model for the survey camp data:

1. There are various features on the UNB campus that surveyed and represented in the CAD files such as Green area, route, and infrastructure features which not matched with the feature classes mentioned above in the table. In Chapter 3, all the feature types on the campus that were represented on the CAD files belonging to the survey collections in years 2011 and 2012 will be discussed.
2. The data type of the feature classes do not meet the framework requirements defined for the survey camp in terms of data spatial representation and tabular behaviour. For example, the ParkingSpace mentioned in the table 2.1 is a point feature class, whereas the parking areas encoded as polylines in the CAD files.
3. The Local Government geodatabase has a large number of datasets and feature classes which mostly are not proper to use for the survey camp data; as a result, in case of using the Local Government geodatabase the management and maintenance processes will be hard and time consuming for the both existing and new survey data.

According to the reasons mentioned above, a geodatabase model produced specifically for the UNB survey camp to support the existing and future surveying data as well as the framework characteristics in order to facilitate the management and maintenance processes. The progress of the geodatabase design and implementation will be discussed with details in the chapter 4 based on the standard design phases and steps mentioned in *Designing Geodatabases* (Arctur and Zeiler, 2004). Table 2.2 illustrates the specifications and advantages of the information model produced in this research project.

Table 2.2: Specifications and advantages of the survey camp geodatabase

Geodatabase aspects	Specifications and advantages
Datasets	<ul style="list-style-type: none"> • The datasets defined and established including: Route, Building_Wall_Countor_Parking, and GreenArea_Infrastructure to organize the survey camp map data in the datasets based on the topological relationships between campus features.
Feature classes	<ul style="list-style-type: none"> • Feature classes were defined in each dataset according to the variety of campus feature represented in the survey camp CAD files. Feature classes are the geodatabase components which store a number of features of a similar type as a single layer (e.g. building, street, tree, or parking lot feature classes). • Additional feature classes created in order to store and manage the data which were not specified in the CAD files such as green field, wooded area, and walkway feature classes. • The spatial data type (point, line, and polygon) of feature classes were determined according to the data content that they should store in the geodatabase. Therefore, feature classes will indicate the variety, content, and type of the campus features. • Feature classes were utilized to create a GIS basemap for the campus. That will help to improve the survey camp data quality in terms of the consistency and validity.
Tabular data	<ul style="list-style-type: none"> • The tabular data represent the features' attribute coding system for the campus features. It supports the survey camp improvement in terms of data consistency as well as data validation and maintenance processes.
Topological rules	<ul style="list-style-type: none"> • The topological rules established between the campus features in order to improve the data consistency and facilitate the data validation process.
Spatial reference system	<ul style="list-style-type: none"> • New Brunswick Double Stereographic (CSRS NAD83) projection is specified as the spatial reference system for the geodatabase feature layers.
Data source	<ul style="list-style-type: none"> • All the spatial data stored in the geodatabase obtained from the CAD files containing survey data of different collection years. • According to the data type and content, the polyline and point layers of the CAD files employed in order to either import or create the spatial data in the geodatabase.

2.3.4 Data Validation and Geodatabase Maintenance

It is critical to validate the data before storing it in the repository. As mentioned, the data tabular behaviour, spatial data type, and the topological relationships between the campus features considered and implemented in building the geodatabase. Accordingly, validation methods are proposed and applied in order to examine the data consistency and correctness with respect to the attribute coding schema and spatial data type of the feature classes. The topology rules are specified and applied as well. The validation methods and topology rules will be explained in the Chapter 4. GIS models were created using the ESRI Model-Builder to maintain the geodatabase. A set of GIS (geoprocessing) tools were utilized to build the GIS models. Each GIS model performs specific GIS tasks to keep the geodatabase updated with new survey camp data. The validation procedure is demonstrated using a sample data in Appendix I. The validation process described below:

1. Identifying the features with incorrect feature codes.
2. Applying the correct feature coding with respect to the proposing coding schema.
3. Converting the CAD files into GIS data, and save them as new feature classes in the geodatabase, and updating the geodatabase with the new feature classes. All of the mentioned processes are performed by the GIS models consisting of a set of GIS tools (geoprocessing tools). The GIS tools were set to perform specific processes from converting the CAD files to GIS data, and updating the layers based on the standard geoprocessing functions (e.g. spatial joins).
4. Applying the topological rules to the feature classes to make sure the features are categorized and stored in the geodatabase correctly.

2.3.5 Web-GIS Service

The Web-GIS service developed in order to publish the spatial and non-spatial content of the survey camp on the web. Moreover, it represents the framework specifications and characteristics with respect to the information system spatial and non-spatial properties and structure. The ArcGIS server was used along with the windows Internet Information Services (IIS) as the GIS server. The actual web GIS service content and functionalities were implemented using ArcGIS JavaScript API. The design and specification of the service will be explained in details in the chapter 5. In bellow, some representative servers are briefly described to demonstrate the research projects that produced web GIS systems using the similar technologies.

- ***Web GIS application with ArcGIS server:*** In this research project the ArcGIS server defined as a sharing objects library of the GIS software. The services published with ArcGIS server contain map, geodata, and geoprocessing. Basic map operations including zooming and panning provided over the forestry data in Linkou Forestry Bureau in China. Furthermore, measurement and query functions established for measuring the distance, showing the XY coordinates, and showing the attribute data such as forest id, name, shape, and area. (Lu H. et al., 2010).

- ***Real state Web GIS application based on ArcGIS server:*** First, a database constructed with attribute and geospatial real state data. Real estate early warning information release platform based on ArcGIS server focuses on sharing and searching information in a distributed environment with multi-user concurrent access (Zhao J. et al., 2010).
- ***Web GIS for Apollo Analyst's Notebook:*** A web GIS developed based on ArcGIS server and ArcGIS JavaScript API to represent the Apollo data. Most of the data was acquired four decades ago during the Apollo mission. Cached map was used and the map was designed to 13 levels of scales to be cached. Data represented as basemap consisting of raster data and feature layers of vector data stored in a file geodatabase. Basic mapping and GIS functions of this WebGIS system include map display, pan, zoom in/out, navigation, and identification. Advanced GIS functions such as special queries can also be submitted to the system to acquire the corresponding results in the map. (Wang J. et al., 2010).
- ***Campus Web-GIS Based on ArcGIS Server:*** Architecture for campus Web-GIS proposed based on ArcGIS server in southern campus of Xidian University. This architecture includes functions of distributing, managing and serving spatial information through internet. The campus information system consists of the ground spatial data including building, road, and green land etc. Distance measuring and layer controlling provided as the major functionalities (Fangli et al, 2010).

The Web-GIS service application publishes the UNB survey camp framework representing the feature layers defined and established in the geodatabase on top of the ESRI World Topographic map as a background basemap. The advanced GIS functionalities established for the service. The functions and tools allow users to explore the map layers and request the features properties using the query function. Furthermore, basic map tools are provided by the service to allow users pan and zoom over the map layers. Feature layers are represented in different zooming levels, and according to their spatial data type. For example, point features such as the trees and manholes are represented in lower zooming levels whereas, the polygonal features such as the buildings.

2.4 Summary

The development process started from examining the CAD files of topographic data and accordingly the framework characteristics defined to establish the systems for minimizing the present and future issues and difficulties. The geodatabase created to categorize, organize, and store the geospatial information based on the conceptual and logical design parameters which in this project particularly tried to meet and satisfy the goal pertaining to the framework characteristics. Data validating methods specified along with the workflow of the geodatabase maintenance. The Web-GIS service developed to represent the framework in terms of visualizing the feature layers and providing the attribute coding system related to each feature type.

Chapter 3 Survey Camp Data Assessment and Framework

Characteristics

3.1 Introduction

Before including survey camp data into a GIS campus database, the suitability of including these datasets was assessed. This chapter describes the results of this assessment and identifies the inconsistencies (i.e. issues) in the survey camp datasets that need to be addressed before including this data in a GIS Campus database. Examples of inconsistencies are shown and an explanation of how these inconsistencies affect the creation and usability of a GIS Campus database is provided. For each issue, this chapter presents a solution as to how a particular issue can be resolved.

These solutions constitute some of the characteristics of the spatial framework. In many cases, these solutions also define a new approach for data collection and identification (topographic surveying), categorization, storage and representation.

3.2 Survey Camp Data Assessment

Prior to developing the framework, survey camp data were analysed and assessed in ArcGIS 10. This assessment of data could help to identify the existing issues regarding data inconsistency. Inconsistency in the data causes the process of storing data in a GIS repository to be hard and time consuming. For this reason, all the present issues were outlined in order to be considered in providing a data validation system. The issues listed below were identified after survey camp data were analysed in ArcGIS 10.

1. Lack of consistency in features' attributes coding system.
2. Inappropriate features encoding with polylines.
3. Incorrect feature attribute coding during either survey camp operation or the mapping process carried out in CAD environment.
4. Undefined shared geometries between the features.
5. Some areal features were not surveyed and mapped as enclosed features.
6. Inconsistent survey of treed and green areas.
7. Inconsistent spatial location of features.

All the issues mentioned above must be considered in the data validation before the survey data could be stored in the GIS repository. Issues will be discussed in details regarding their effects along with examples indicating the problems in different cases.

3.2.1 Lack of Consistency in Features Attributes Coding System

Attributes are critical factors in developing an efficient and appropriate GIS system. They allow faster and easier data identification and retrieving. In survey camp data, there is no consistent feature attribute coding system. This problem causes the process of data storing and retrieving to be highly dependent on manual editing. For example, one group may code buildings as “B” whereas another group may not use any specific attribute code. Figure 3.1 and 3.2 show examples of this issue.

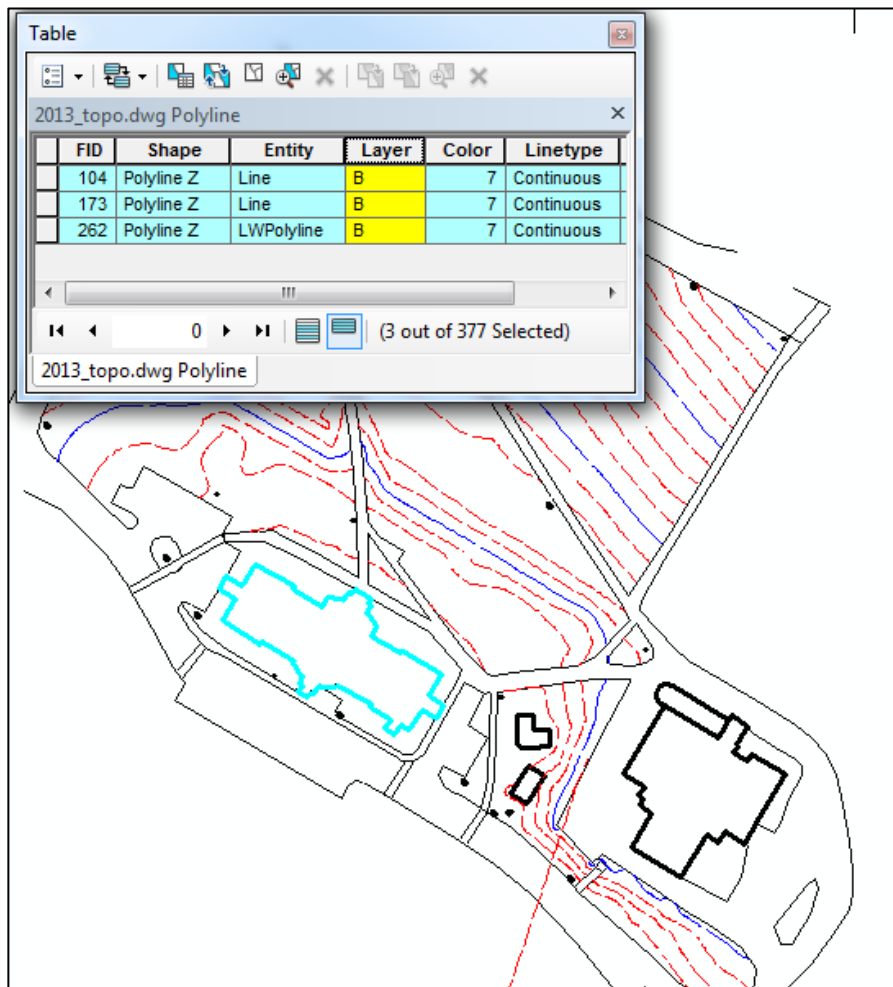


Figure 3.1: Example of a building feature coded as “B”

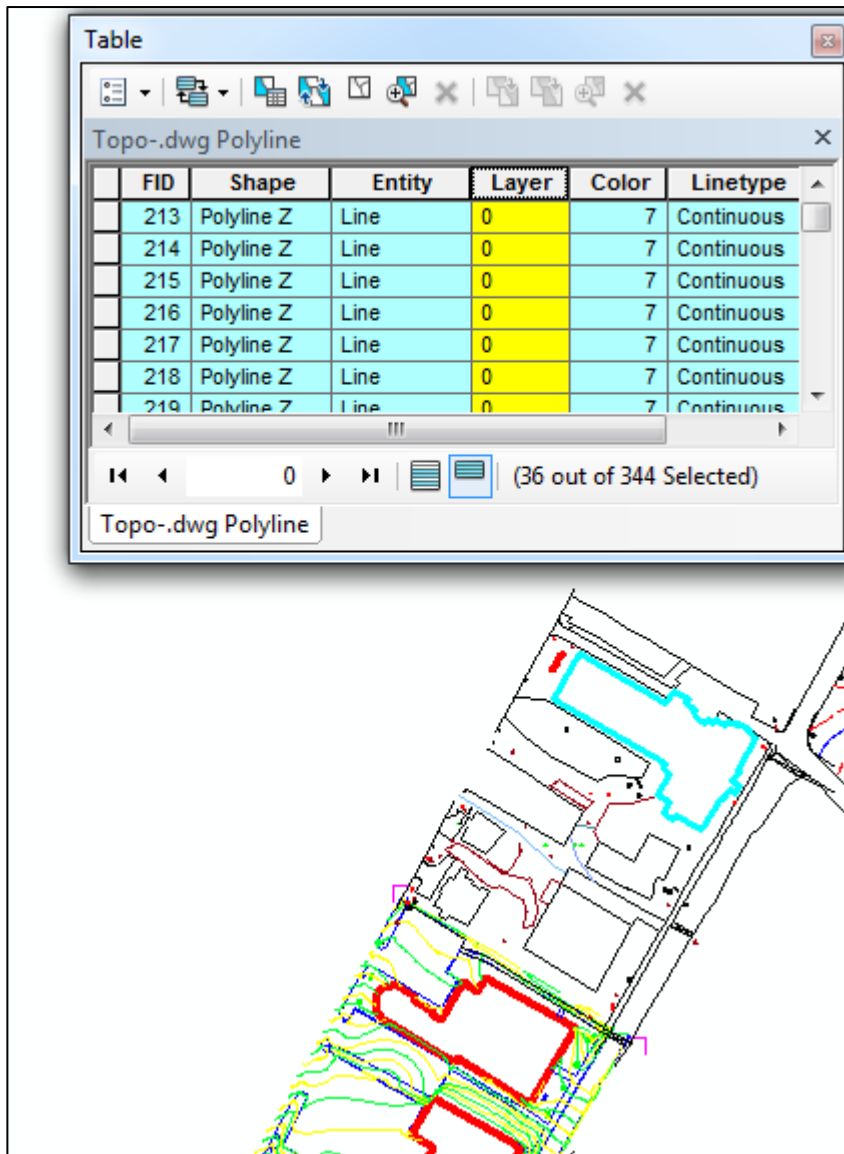


Figure 3.2: Example of a building feature with no specific code

3.2.2 Inappropriate Feature Encoding

Polylines were used in the CAD environment to generate the campus topographic maps from survey points. Survey points contain coordinates of different types of features on the campus. However, all those different features were mapped only with polylines. Whereas, in the GIS environment point, line, and polygon data types employed to map and store the features with regard to their spatial characteristics. Having features encoded with different spatial representation types (point, line and polygon) is an advantage of a GIS system. This advantage helps the users to provide advanced cartographic products.

Figure 3.3 shows a section of campus in ArcMap. There are various spatial objects in this section such as buildings, parking lots, sidewalks and streets which all spatially represented with polylines. In a GIS environment, these features would be interpreted better if they had been represented with a combination of lines and polygons instead of only polylines. Better interpretation of map objects highly facilitates the process of managing the objects in the GIS repository.

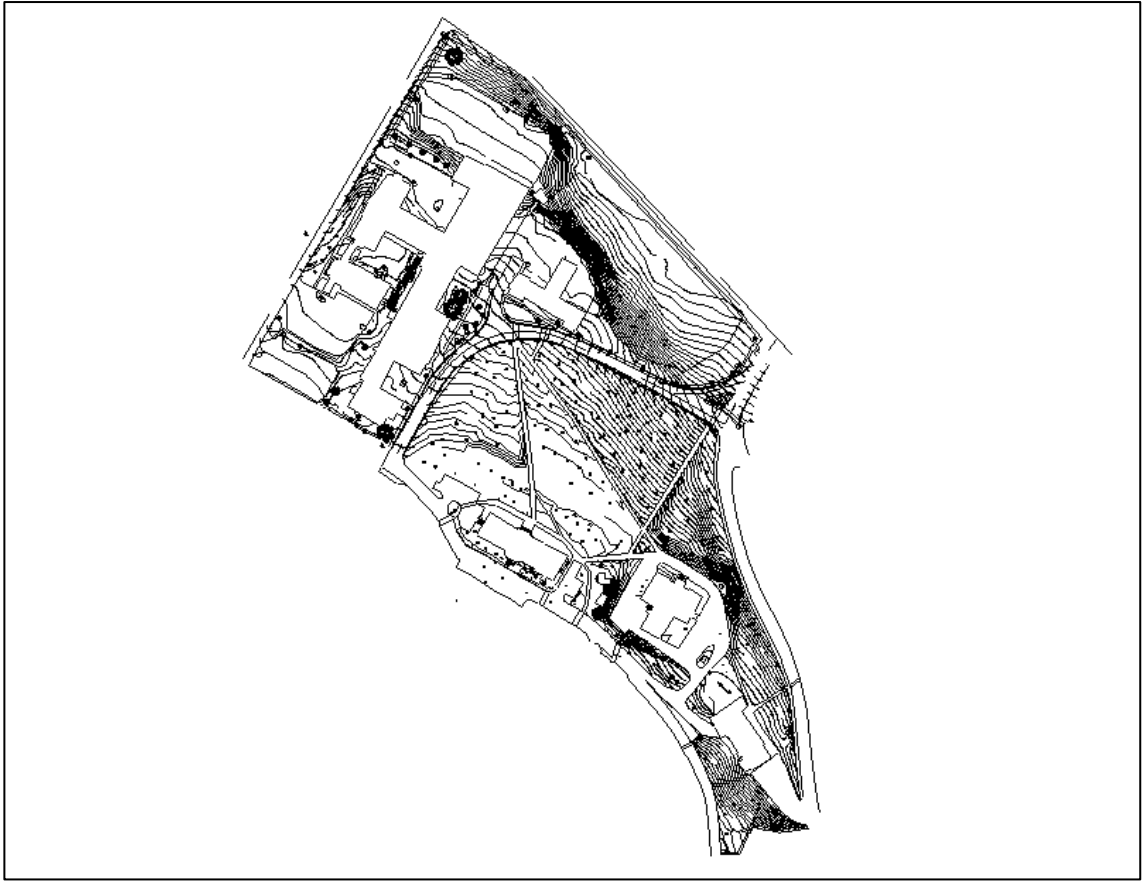


Figure 3.3: Example of the features encoded with polylines

3.2.3 Incorrect Feature Attribute Coding

Different features were often incorrectly coded during the survey camps. It is shown in figure 3.4 that objects with “Buildings” attribute codes were selected but other types of features (street center lines) were selected as well. As mentioned, these coding issues have large effect on the time required to manage, validate, and finally store the data due to the incorrect interpretation and identification of maps’ objects.

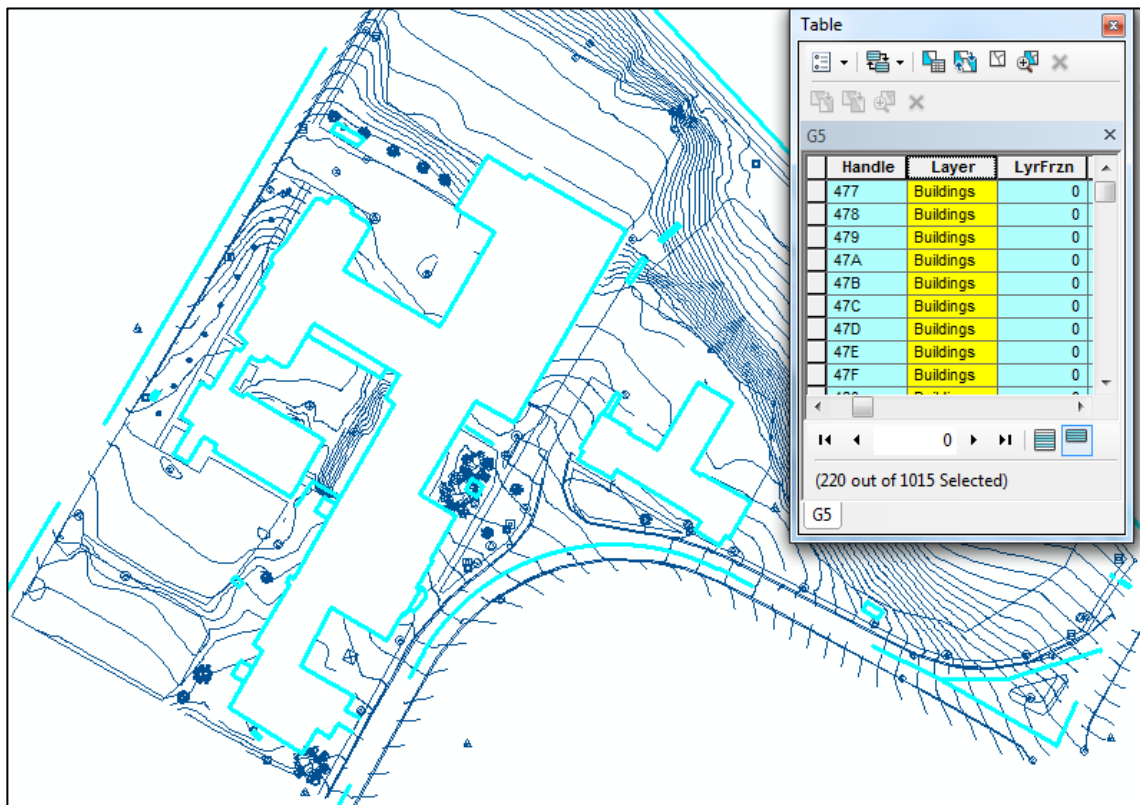


Figure 3.4: Example of center lines coded incorrectly as buildings

3.2.4 Undefined Shared Geometries

Shared geometries between different features were not defined in the original CAD maps. For example, edges (or boundaries) between the buildings on the university campus and the parking lots are not defined as being shared. Figure 3.5 shows an example in which the shared boundary between a building and a parking lot is coded as building.

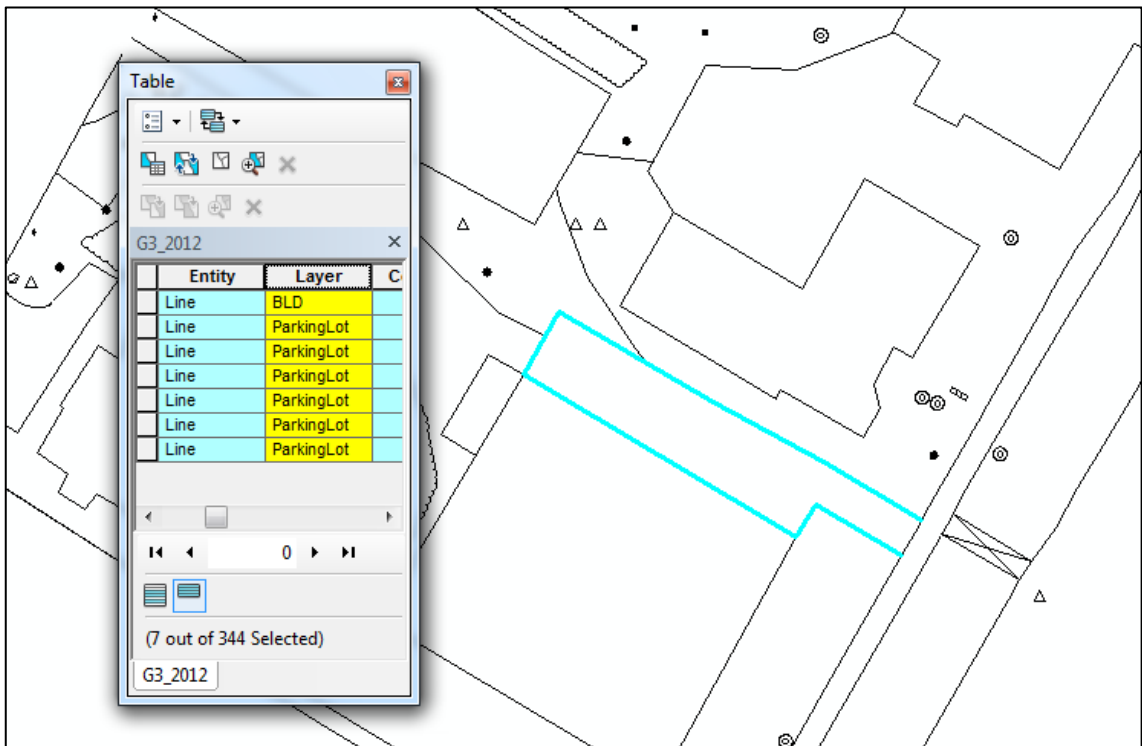


Figure 3.5: Example of incorrect coding for the shared geometries between the buildings and parking lots

3.2.5 Unenclosed Areal Features

Some areal features such as parking lots and green areas were not surveyed and mapped as enclosed features. That is, there is a gap between the starting and ending points of these features. Figure 3.6 shows two gray polygonal features which represent parking lot areas. These two polygons are examples of the expecting results of mapping parking lot features after storing in the GIS repository. In contrast, the highlighted lines show the parking lot polylines of the original survey camp CAD files were not surveyed and mapped as enclosed features.

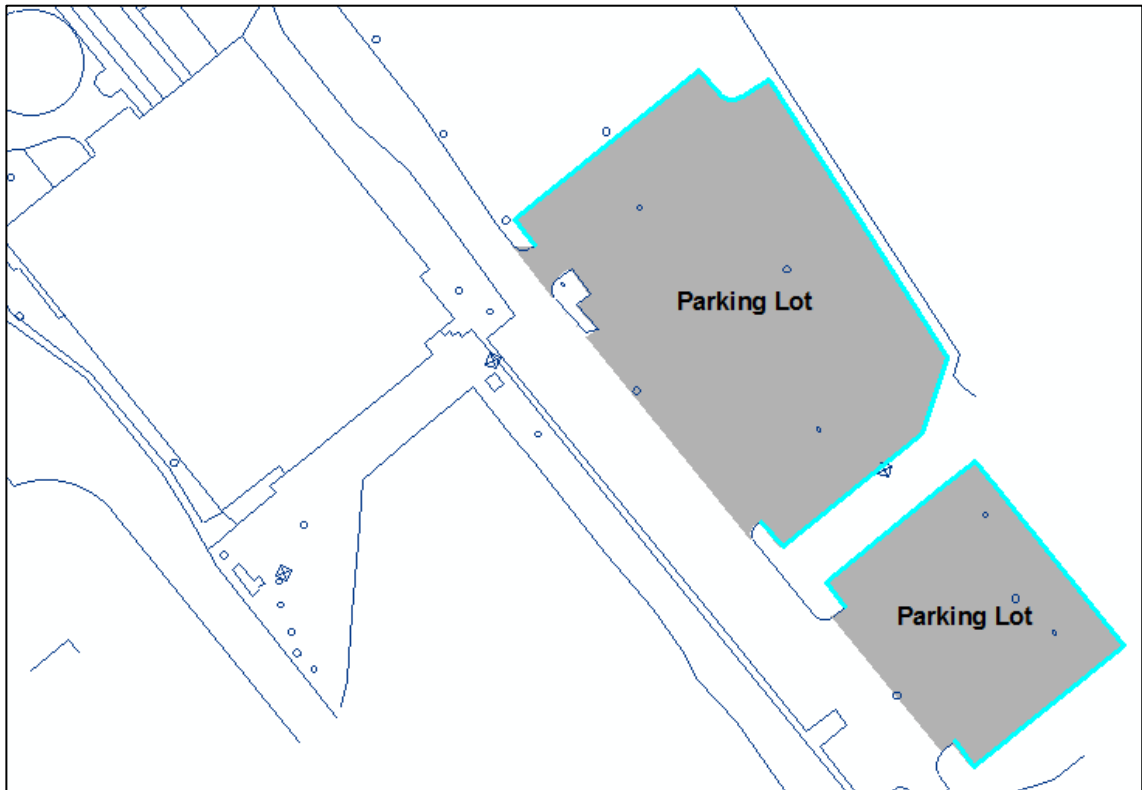


Figure 3.6: Comparison of the parking lot polygons with the CAD parking lot polylines

3.2.6 Inconsistent Survey of Treed Areas and Green Fields

Some groups delineated treed areas whereas other groups did not. There are various areas in the campus which are generally identified as green areas. However, some of these areas are treed areas as well and not just the green areas. In survey camp, trees are surveyed and represented in the CAD maps mostly with the symbols including the shapes that formed with lines. The symbols; in spite of being appropriate for mapping, do not signify the exact locations of the trees. On the other hand, the point layers of some CAD files do not provide enough data and the source of the tree features (symbols) on the maps were not available.

Although, only in few CAD maps, there are areas that contain a sufficient number of trees and are mapped as treed or wooded areas. On the other hand, there are some green areas which contain few tree points. These areas are rather to survey as a green field along with the tree points within the fields. Two maps of two different survey camp groups are shown in figure 3.7. In those maps there are selected points and lines which respectively represent trees' points in one map and a wooded area in another group's map. In this case, like the wooded area, it is rather the areas which contain the trees' points to be surveyed and mapped as enclosed green fields. However, surveying the green fields as enclosed objects doesn't imply that trees' points must not be surveyed within those areas.



Figure 3.7: Example of the trees' points and the enclosed lines of the wooded area

3.2.7 Inconsistent Spatial Location of Features

Inconsistency occurred in spatial location of features. Different surveys often show the same feature in different spatial location. The survey points are collected initially based on local coordinate systems. After completion of surveying, the collected points are transferred to New Brunswick Double Stereographic (NAD83 CSRS) projection. The transformation is implemented based on the control points which are established using GPS observations.

In some cases, the features in the CAD maps have different spatial location in comparison with the corresponding features from another year of survey. This issue could occur due to the inappropriate GPS observation and respectively inaccurate transformation of coordinate systems. However, distance measurements between the corresponding features of survey datasets of various years indicate that survey data are relatively accurate. This relative accuracy indicates that features were surveyed appropriately.

Assessing the survey data accuracy is outside the scope of this project, that, therefore no method was defined to validate the data in terms of accuracy. However, in order to partially solve this issue and maximize the consistency, the most recent survey data were used to create and populate the datasets in the repository.

3.3 Framework Characteristics

A set of characteristics have been proposed below in order to solve the inconsistency problems outlined as the results of the data assessment. The survey camp data storage and validation processes in the GIS repository will be facilitated applying the framework characteristics to:

- 1- Provide consistency in the feature surveying
- 2- Establish common feature attribute coding schema
- 3- Define the spatial data type of the features
- 4- Identify the shared geometries
- 5- Define the spatial reference system for the features

Each of the characteristics mentioned above will be described in details. It will be described how each of these characteristics will solve the problems and minimize the inconsistency issues. As a matter of fact, these factors are to be considered during the land survey operations and mapping procedures. Based on these characteristics, we can support the survey data in terms of consistency and validation to be used and organized in a GIS system. By following these, we can assure the GIS repository will be correctly and easily maintained. Furthermore, the characteristics must be published via the Web-GIS service to apply in the survey camp by students.

3.3.1 Consistent Feature Surveying

Features to be surveyed must be constant and consistent for all areas on the university campus. There are many different types of features and areas all around the campus and students' CAD maps represent many of those features. Therefore, one of the critical framework characteristics is to establish a consistent feature surveying system by defining the key spatial features and areas of the campus. This system or characteristic will support the survey camp in terms of data completeness and consistency. For example, all survey groups must survey a treed area as a treed area and a green field as a green field. In this case, the web service will allow the users to interpret the key map layers and accordingly the users will identify the key areas and features of the campus. In other word, based on the published map layers on the web service, students will know what types of features must be surveyed on the campus.

Prior to produce a system to support the consistency in feature surveying, it is essential to establish a system for feature coding as well. The feature coding system will fulfill the goals of identifying the key spatial features. Along with the feature coding system, the various features which are expected to be surveyed will be specify and discussed in section 3.3.2 .

3.3.2 Feature Coding System

A common feature coding system will be used to identify the features; For example, all buildings will be coded as “BL”. 2012 and 2011 CAD data were studied in order to provide a common coding system which can cover majority of the features in the campus. As a result, feature codes used in 2012 and 2011 data were summarized in the table 3.1 and then used in a reclassification process. The reclassification process resulted in establishing a new system of feature attribute coding. The new coding system is to be used in the survey operation as IDs for the surveying points of the objects and features.

Moreover, the point IDs must be applied in the mapping processes implemented in CAD software to indicate the names of the feature layers. Therefore, it is easier in the GIS repository to convert, edit, and finally store the data obtained from the CAD maps containing the feature layers that coded according to the framework coding system. Like the feature layers, the feature codes will be represented in the web service. Collected survey data in years 2011 and 2012 covers a sufficient area of the campus that can be seen in the figure 3.8 as six different sets of survey maps. Moreover, in terms of the data completeness and having up-to-date data these years are more reliable than the others.

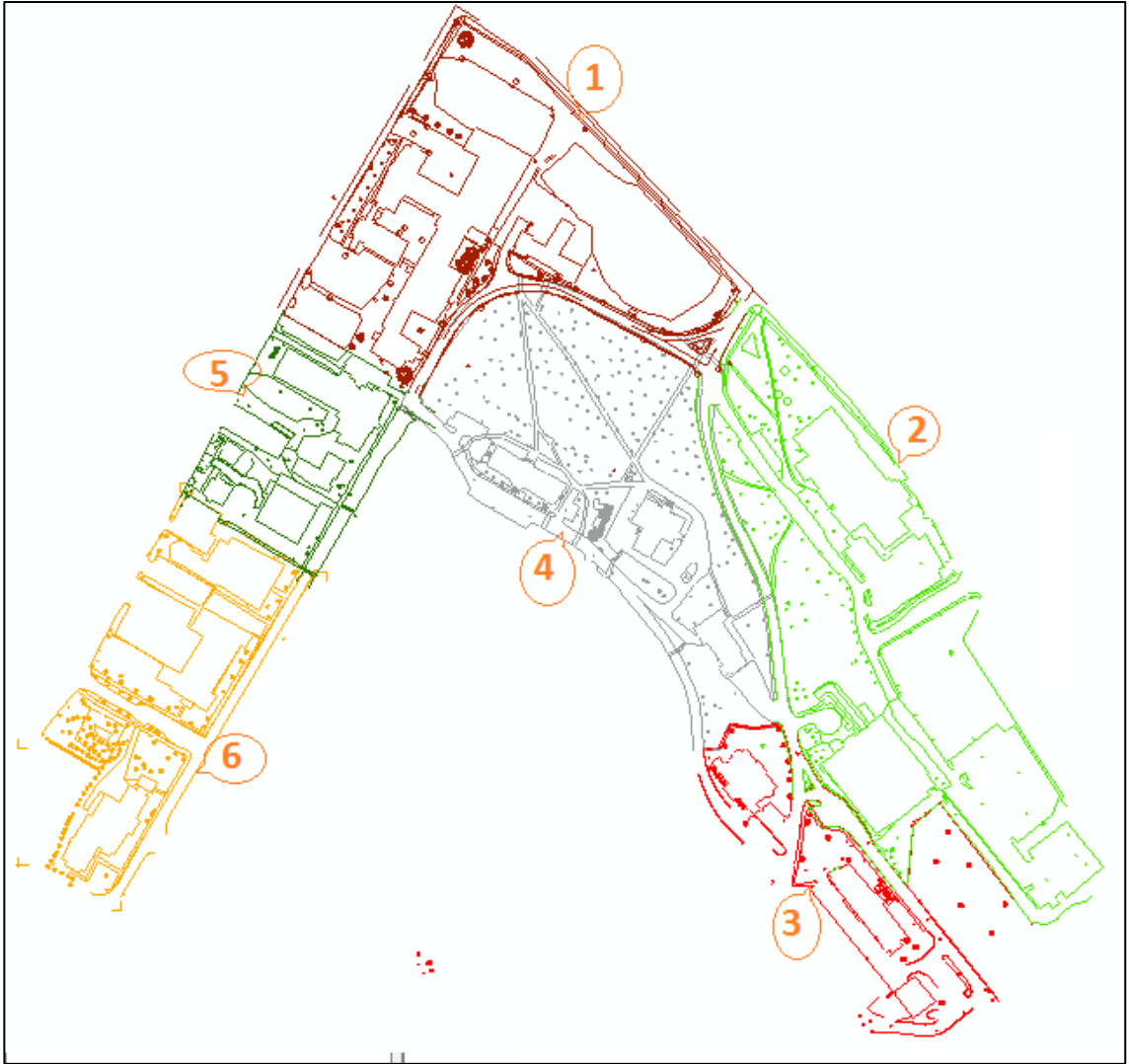


Figure 3.8: Map sets of the survey camp of years 2012 and 2011

Table 3.1: Summary of the feature codes of the map sets of years 2012 and 2011

	Map Set 1	Map Set 2	Map Set 3	Map Set 4	Map Set 5	Map Set 6
1	Asphalt	Building	Buildings	Benches& Bikerack	BLD	Asphalt
2	Buildings	contours	Contour	Buildings	Electrical box	Block
3	Contours	Fence	Road	Contours	Hydrant	Building
4	Curb	Hydrant	Tree	Curbs&walkways&sidewalk	Parking lot	Contours
5	Lamp Post	Light	VPORT	Manholes & Firehydr & sign	Sidewalks	Fence
6	Parking Lot	Retaining Wall	-	Tree&Post	Signs	RWALL
7	Sidewalk	Road&Sidewalk	-	-	Telephone poles	Sidewalk
8	Sign	Storm Drain	-	-	Tree Line	Top of curb
9	Tree	Vegetation	-	-	VPORT	Tree
10	VPORT	-	-	-	-	-
11	Walkway	-	-	-	-	-
12	Woods	-	-	-	-	-

In the table 3.1, some map sets such as the map set 1 have an appropriate number of classes of the feature codes. In contrast, the other map sets such as map set 4 and 3, have only few number of code classes.

Additionally, there are some codes such as those mentioned in row 4 of the map set 4 which indicate to three different types of features; whereas, those features are incorrectly identified as one in a GIS system due to the wrong identification attribute code. Now, according to the all the records in table 3.1 we must establish a common feature coding system. The new attribute coding system employed to:

1- Provide appropriate number of code classes which are to cover all the classes listed in table 3.1.

2- Provide feature codes with only two letters to use more conveniently during the surveying and mapping procedures. There is an exception for contour lines. Contour lines are generated from elevation points and are stored in contour layers by default in CAD software. As a result, all the contour lines are available in the CAD maps as contours. However, elevation points can be collected and coded with any appropriate coding system.

Table 3.2 shows the new proposed coding system. Additionally, the codes obtained from summarization of the six map sets are listed in the table 3.2 to show how new codes were generated by reclassifying the students' codes. The number of new code classes also specifies the number of different feature types which students should consider in the surveying collections.

Table 3.2: Summary of the campus features attribute codes

New feature coding system	Feature type	Students' feature codes
BL	Buildings	<ul style="list-style-type: none"> • Buildings • BLD
BN	Benches	<ul style="list-style-type: none"> • Benches
BR	Bike racks	<ul style="list-style-type: none"> • Benches&Bikerack
Contours	Contour lines	<ul style="list-style-type: none"> • Contours
CB	Curbs	<ul style="list-style-type: none"> • Curb • Top of curb • Curbs&walkways&sidewalk
EB	Electrical boxes	<ul style="list-style-type: none"> • Electrical box
FN	Fences	<ul style="list-style-type: none"> • Fence
GF	Green fields	<ul style="list-style-type: none"> • Vegetation
HD	Hydrants	<ul style="list-style-type: none"> • Hydrant • Manholes&Firehydr&sign
LI	Lights	<ul style="list-style-type: none"> • Light
LP	Lamp posts	<ul style="list-style-type: none"> • Lamp Post • Tree&Post
MH	Manholes	<ul style="list-style-type: none"> • Manholes&Firehydr&sign
PG	Play grounds	-
PL	Parking areas	<ul style="list-style-type: none"> • Parking Lot
SI	Signs	<ul style="list-style-type: none"> • Sign • Signs
ST	Streets	<ul style="list-style-type: none"> • Asphalt • Road • Road&Sidewalk
SD	Storm drains	<ul style="list-style-type: none"> • Storm Drain
SW	Sidewalks or any type of walkways	<ul style="list-style-type: none"> • Sidewalk • Road&Sidewalk • Curbs&walkways&sidewalk
WA	Treed or wooded areas	<ul style="list-style-type: none"> • Woods • Tree Line
TR	Trees	<ul style="list-style-type: none"> • Vegetation • Tree • Tree&Post
RW	Retaining Walls	<ul style="list-style-type: none"> • Retaining Wall • RWALL
TP	Telephone poles	<ul style="list-style-type: none"> • Telephone poles

3.3.3 Spatial Representation Forms

All the features of the same type must have a consistent spatial representation form. For example, the parking lots must be surveyed as enclosed features as opposed to some being surveyed as enclosed features and others not. This factor will highly support the advanced visualizing, identification, and cartographic results. As the outcomes of the survey camp are the vector data including the topographic maps; thus, these data must be represented with points, lines, and polygons considering the feature types indicating by the data. Table 3.3 shows the preferred spatial representation regarding the various features on the campus with respect to the listed feature types in table 2.

Table 3.3: Geometry types of the campus features

Point features	Line features	Polygonal features
<ul style="list-style-type: none">• Electrical boxes• Hydrants• Lights• Lamp posts• Manholes• Signs• Storm drains• Trees• Telephone poles	<ul style="list-style-type: none">• Contour lines• Curbs• Fences• Streets• Sidewalks• Walkways• Retaining walls	<ul style="list-style-type: none">• Buildings• Benches• Green fields• Parking areas• Playgrounds• Wooded areas

3.3.4 Shared Feature Types

Shared feature types must be indicated in the surveyed data. The coding system can be useful in this part as well. For example, buildings and parking lots, streets and parking lots, or streets and sidewalks can have shared geometry. Therefore, shared geometry must be coded in a way to indicate the boundaries shared between some different types of features. Additionally, specifying the shared geometry will allow some specific features to be surveyed as enclosed features such as parking lots, which have shared geometry with most of the other features. Table 5 shows some cases for the shared geometry from 2012 CAD files.

Tale 3.4: Examples of the features with shared geometry

Features with shared geometry	Coding system for shared geometry
<ul style="list-style-type: none"> • Buildings-Parking areas • Buildings-Green fields • Buildings-Streets 	<ul style="list-style-type: none"> • BL-PL/PL-BL • BL-GF/GF-BL • BL-ST/ST-BL
<ul style="list-style-type: none"> • Green fields-Parking areas • Green fields-Streets • Green fields-Sidewalk/Walkways 	<ul style="list-style-type: none"> • GF-PL/PL-GF • GF-ST/ST-GF • GF-SW/SW-GF
<ul style="list-style-type: none"> • Parking areas-Sidewalks • Parking areas-Streets 	<ul style="list-style-type: none"> • PL-SW/SW-PL • PL-ST/ST-PL

3.3.5 Spatial Reference System

Since the survey camp data was transferred from a local coordinate system to New Brunswick Double Stereographic (NAD83 CSRS) projection, this spatial reference system is used for the GIS system.

3.4 Summary

In order to develop the spatial framework by designing a GIS system first, the survey data and maps were analyzed to identify the inconsistency and incompatibilities problems. After identifying the existing inconsistencies in the topographic maps, the framework requirements and solutions were specified and proposed first to minimize the difficulties and issues and then to optimize the time and processes required to manage the geospatial information in the GIS repository. Applying the solutions will significantly reduce the data inconsistencies; thus, a faster and easier data validation, storage, and usage will be experienced in the system. As a fact, identifying the problems and outlining the possible solutions were the most important aspects to mention in this chapter prior to designing the repository. All the requirements mentioned here considered in developing the web service as well. In fact, the web service designed with respect to the framework requirements in order to represent a model of the framework to the students.

Chapter 4 Geodatabase

4.1 Introduction

This chapter outlines the procedure of designing and building a geodatabase for managing student surveyed topographic data of UNB Campus. After building the geodatabase, the process of loading the data will be described. The geodatabase will be evaluated, and feature layers will be represented on the maps with scales of 1:6000 and 1:2500. These scales can show the campus area in a full extent, as well as the point features around the campus respectively. Along with representing the final results, the methods and workflow of maintaining the geodatabase will be explained. Furthermore, method of documenting the geodatabase is illustrated and then the geodatabase schema will be documented. An example is given to partially illustrate the schema documentation. The full geodatabase schema documentation has been shown in Appendix II as well.

The chapter continues with explaining the data validation methods and building the GIS models. The GIS models are illustrated in terms of the functionalities and analyzing tools. Besides, appropriate methods for validating the data are outlined.

4.2 Geodatabase Design Phases

This section describes the procedure of designing a geodatabase in order to organize and store the survey camp topographic data. As mentioned in *Designing Geodatabases* [Arctur and Zeiler, 2004], three key sections including conceptual, logical, and physical design phases provide guidance to create a dynamic GIS data model. The design phases consist of the 10 steps including modeling the users' view, defining the objects and relationships, selecting geographic representation, matching geodatabase elements, and organizing the geodatabase structure. In the conceptual design, the thematic layers identified and characterized with respect to the framework requirements and applications. It is essential to specify the spatial representation, usage, scale, symbol, and the data source while identifying the thematic layers [Arctur and Zeiler, 2004]. Characterizing the thematic layers resulted in initial specification of the geodatabase design elements. In this project, those elements comprised the datasets, feature classes, topologies, and the features' attribute codes and domains.

In the logical design phase, the spatial and non spatial specifications and properties of the data in the GIS repository specified and the final design proposed. The final phase is the physical design in which the actual geodatabase created with respect to the spatial and non spatial specifications defined in the first two phases. In the physical design, data loaded into the feature classes and organized in the feature datasets. Topological rules applied to the datasets and tabular data arranged.

Finally, the design documented to illustrate the GIS data model and the specifications of the geodatabase. Table 3.2.1 illustrates the 10 steps included in the design phases mentioned above.

Table 4.1: Phases and steps of designing a geodatabase (Arctur and Zeiler, 2004)

Design Phases	Steps
Conceptual Design	<ol style="list-style-type: none"> 1. Identify the information products produced with the GIS system 2. Identify the key thematic layers 3. Specify the scale range and spatial representation for each thematic layers 4. Group representations into datasets
Logical Design	<ol style="list-style-type: none"> 5. Define the tabular structure and behavior of descriptive attributes 6. Define the spatial properties of the datasets 7. Propose a geodatabase design
Physical Design	<ol style="list-style-type: none"> 8. Implement the design 9. Design workflows for building and maintaining each layer 10. Design Documentation

Each phase will be explained individually in later sections along with the steps mentioned in the table above. However, steps 2 and 3 of conceptual design will be combined and explained in section 4.3.2 to outline the thematic layers specifications.

4.3 Conceptual Design

The main purpose of the conceptual design phase is to characterizing the survey camp key thematic layers. Thus, the results of the survey camp CAD files assessment and the framework characteristics, as described in Chapter 3, are significantly helpful in this phase. For example, all the mentioned inconsistency issues regarding feature attribute coding, feature encoding, shared geometry, etc. must be considered in the design. Furthermore, the outlined spatial framework characteristics particularly those that indicate the features coding system and features spatial representation, significantly facilitate the process of identifying the thematic layers, specifying layers' spatial properties, and organizing the datasets.

4.3.1 Information Products

As mentioned in the workfellow and solution and illustrated in figure 2.1 the information products outlined in below are produced for the GIS system and with respect to the survey camp and spatial framework requirements.

1. A GIS repository for UNB Campus which contains survey camp data. The repository contains the best survey camp data in terms of the year of data collection as well as the data completeness.
2. A Web-GIS service which represents the information in the GIS repository.
3. The Documentation that illustrates the key design and verifies the specifications of the designed geodatabase regarding the created datasets, domains, and defined rules.

4.3.2 Thematic Layers Specifications

In this step the thematic layers identified according to the feature types listed in tables 4.2 and 4.3. Each thematic layer will be identified in terms of the map use, data source, representation, spatial relationship, map scale, and symbology. Tables 4.2 and 4.3 show the specifications of each thematic layer.

Table 4.2: Specifications of the campus key thematic layers

Layer	Specifications	Description
Building	Map use	Shows buildings footprints
	Data source	Building polylines on the CAD maps
	Representation	Polygons
	Spatial Relations	Buildings polygons must not overlap
	Map scale	Visible at all scales
	Symbology	Polygons with fill color and outlines
Contours	Map use	Shows contours lines
	Data source	Contour layers on the CAD maps
	Representation	Lines
	Spatial Relations	Must not cross, must not intersect with buildings
	Map scale	Visible at 1:3000 and larger
	Symbology	Contour line symbol
Electrical facilities	Map use	Represents all types of electrical features
	Data source	Points of the electrical features on the CAD maps
	Representation	Points
	Spatial Relations	Must be inside the green fields polygons
	Map scale	Visible at 1:3000 and larger
	Symbology	Each type of electrical features symbolized with different points symbols
Green Areas	Map use	Represent all types of green and wooded areas
	Data source	Wooded areas and green fields polylines on the CAD maps
	Representation	Polygons for the green fields and wooded areas
	Spatial Relations	Trees and electrical facility points must be inside the green field polygons
	Map scale	Visible at all scales
	Symbology	Point symbols and fill colored polygons

Table 4.3: Specifications of the campus key thematic layers

Layer	Specifications	Description
Parking Areas	Map use	Shows all types parking lots and Bike racks
	Data source	Parking lots and bike rack polylines
	Representation	Polygons
	Spatial Relations	Must not overlap with buildings and bike racks
	Map Scale	Visible in all scales
	Symbology	Fill colored polygons
Signs	Map use	Shows the signs on the campus
	Data source	Point features indicating signs in CAD files
	Representation	Points
	Spatial Relations	Must be inside the green fields polygons
	Map Scale	Visible at 1:3000 and larger
	Symbology	Point symbols
Streets	Map use	Represents the streets and curbs on the campus
	Data source	Street polylines on the CAD files
	Representation	Lines
	Spatial Relations	Streets must not intersect other lines of similar layer. Streets must not intersect walkways.
	Map scale	Visible in all scales
	Symbology	Different lines symbology
Hydrants& Storm water facilities	Map use	Shows the hydrants and storm water facilities
	Data source	Manholes, hydrants, and drain storms points on CAD files
	Representation	Points
	Spatial Relations	Must be inside the green fields
	Map scale	Visible at 1:3000
	Symbology	Different point symbols
Walking route	Map use	Shows all types of walking routs on campus
	Data source	Walk way, Sidewalk, and cross walk polylines on the CAD maps
	Representation	Lines
	Spatial Relations	Walkways must not intersect with streets
	Map scale	Visible in all scales
	Symbology	Line symbols
Walls	Map use	Shows walls and fences on the campus
	Data source	Walls and fence polylines on the CAD maps
	Representation	Lines
	Spatial Relations	Must not intersect other lines with similar layer
	Map Scale	Visible in all scales
	Symbology	Line symbols

4.3.3 Datasets Specifications

Generally “dataset” is a generic term that refers to a collection of data (e.g. survey camp 2011 dataset of Head Hall area). In this chapter, dataset refers to a group of feature classes (refer to Appendix I.4 Glossary of GIS terms). Datasets are utilized to group the feature classes that share spatial reference, spatial relationships and topologies.

Feature class refers to a set of features that have a common geometry type (point, line, and polygon). A feature class in a geodatabase typically contains geographical objects of the same type in terms of geometry and content (e.g. building feature class contains buildings, street feature class contains roads, etc.).

Each feature class indicates a specific feature on the campus that was used to define and create the thematic layers. However, because the survey camp data are collected every year; thus, the feature classes should contain the best available data in terms of providing the most recent year of collection as well as the most complete data. Table 4.4 shows the datasets along with the feature classes and topologies included in each dataset.

Table 4.4: Datasets and feature classes

Buildings_ Wall_ Contours_ Parking feature dataset			
Buildings	Polygon feature class	Fence	Line feature class
Bick racks	Line feature class	Guardrails	Line feature class
Contours	Line feature class	Retaining walls	Line feature class
Building_ Wall_ Contour	Geodatabase topology	Parking lots	Polygon feature class
Green Area_ Infrastructure feature dataset		Route feature dataset	
Benches	Polygon feature class	Crosswalks	Line feature class
Electrical boxes	Point feature class	Curbs	Line feature class
Green fields	Polygon feature class	Sidewalks	Line feature class
Hydrants	Point feature class	Streets	Line feature class
Lamp posts	Point feature class	Walkways	Line feature class
Manholes	Point feature class	Route	Geodatabase topology
Playground	Polygon feature class	-	-
Signs	Point feature class	-	-
Storm drains	Point feature class	-	-
Telephone poles	Point feature class	-	-
Trees	Point feature class	-	-
Wooded areas	Polygon feature class	-	-
GreenArea_ Infrastructure	Geodatabase topology		

4.4 Logical Design

In the logical design phase, the non spatial and spatial behaviour of the feature classes were defined. Valid codes were specified for the features as the coded value domain and the topological rules were established in each dataset with respect to the spatial relationships between the feature classes. The codes and rules specified for the features will allow for validating. Student surveyed data during the data loading. This validation will both identify inconsistencies in the data (and thus allow for correction) and ensure that the data in the geodatabase adheres to appropriate specifications.

4.4.1 Define the Tabular Structure and Behavior

In this step of the logical design the domains specified for the geodatabase and the valid codes as defined for each individual feature identified in the earlier steps is noted. The coded value domain prepared in order to establish a system to validate the survey camp feature attributes. The valid codes obtained from the feature coding system that mentioned as one of the framework characteristics. Moreover, the coding system produced regarding the shared geometry considered and added in the domain as well. Thus, the domain will allow confirming the input geospatial data in terms of the consistency and validity of the attributes. The domain's coded value listed in table 4.5 along with the descriptions with respect to the type and class of each feature.

Table 4.5: Geodatabase coded value domains

Code	Description	Code	Description
BL	Building	TP	Telephone pole
BS	Building stair	VP	Vport
BN	Bench	WA	Wooded area
Contours	Contours	WW	Walk way
CB	Curb	GFST	Green field-Street
CW	Cross walk	GFWW	Green field-Walkway
EB	Electrical Box	PLST	Parking lot-Street
FN	Fence	PLSW	Parking lot-Sidewalk
GF	Green Field	BLPL	Building-Parking lot
GD	Guardrail	BLGF	Building-Green field
HD	Hydrant	GFPL	Green field-Parking lot
LI	Light	GFSW	Green field-Sidewalk
LP	Lamp post	STGF	Street- Green field
MH	Manhole	WWGF	Walkway- Green field
PL	Parking lot	STPL	Street- Parking lot
RW	Retaining wall	PLBL	Parking lot- Building
SI	Sign	GFBL	Green field- Building
SR	Stairs	STBL	Street- Building
ST	Street	PLGF	Parking lot- Green field
SD	Storm drain	SWGf	Sidewalk- Green field
SW	Sidewalk		
TR	Tree		

After establishing the domain for the geodatabase, it is essential to determine that which attribute field of the feature classes must be associated with the domain. Since all the CAD files have an attribute field as *Layer* which contains the features' codes of interest, the Layer attribute field should be associated with the domain established for the geodatabase.

4.4.2 Define the Spatial Properties of the Datasets

Since the survey camp data is in New Brunswick Double Stereographic projection (NAD83, CSRS), this spatial reference system was set for all the feature datasets. The spatial reference defined for the datasets is assigned to each feature class that created in the datasets. This datasets advantage prevents storing a feature class with a different spatial reference in the geodatabase.

Topological rules are the next spatial properties to set for the datasets. The topological rules established according to the logical spatial relationships between the features of similar or different classes. For example; logically, contour lines must not intersect or overlap with each other or with the retaining walls. Moreover, there are various types of features specifically expecting to be spatially related with each others in the real world. For example signs, electrical facilities, and some water facilities are always located in the green fields; as a result, it is necessary to confirm if, those features are located and identified correctly on the survey camp topographic maps in terms of the spatial relationships. Tables 4.6, illustrates the topological rules in each dataset.

Table 4.6: Topology rules

Building_Wall_Contour_Parking dataset		
Origin feature class	Topological rules	Comparison feature class
Building	Must not overlap	-
Building	Must not overlap with	ParkingLots
Building	Must not overlap with	Bick racks
Contours	Must not self-intersect	-
Contours	Must not intersect with	BuildingBoundaries
Contours	Must not intersect with	RetainingWall
Contours	Must not intersect	-
ParkingLots	Must not overlap	-
ParkingLots	Must not overlap with	BickRacks
GreenAreas_ Infrastructure dataset		
GreenFields	Must not overlap	-
WoodedAreas	Must not overlap	-
Hydrants	Must be properly inside	GreenFields
Signs	Must be properly inside	GreenFields
Trees	Must be properly inside	GreenFields
Benches	Must be covered by feature class of	GreenFields
TelephnePoles	Must be properly inside	GreenFields

4.4.3 Propose a geodatabase design

A geodatabase for UNB campus was proposed according to the mentioned specifications in the conceptual and logical design phases and also with respect to the survey camp requirements and the stakeholders associated with this project. After all, the proposed designed were applied in building the geodatabase using appropriate tools and methods which will be described in the physical design phase. Appendix II shows the full documentation of the geodatabase schema.

4.5 Physical Design

In the physical design phase, the procedure of building the geodatabase will be described in terms of applying the specifications identified in conceptual and logical design phases. In this phase, the geodatabase was built and then the data were loaded to the corresponding datasets and feature classes. The valid codes were associated to the features' attributes and the spatial rules were applied to the feature classes as well. Furthermore, examples were given to illustrate the procedure of creating the geodatabase and to visualizing the outcomes associated with the data stored in the geodatabase.

4.5.1 Implement and Prototype Geodatabase Design

As mentioned in *Designing Geodatabases* [Arctur and Zeiler, 2004] there are six steps to create and populate a geodatabase. The first step includes the options of starting the building procedure. Since in this project the geodatabase were specifically designed to organize the UNB survey camp data, the option of creating an empty geodatabase in ArcCatalog was selected as a starting point. In the second step, an empty personal geodatabase was created in ArcCatalog by properly applying the specifications mentioned in the logical design. The data were loaded into the geodatabase in the third step and then the topologies were applied in the step four. The model then was tested in terms of the performance and outcomes. Finally the model was revised and corrections applied as needed. Figure 4.1 illustrates the steps and the workflow of implementing the geodatabase.

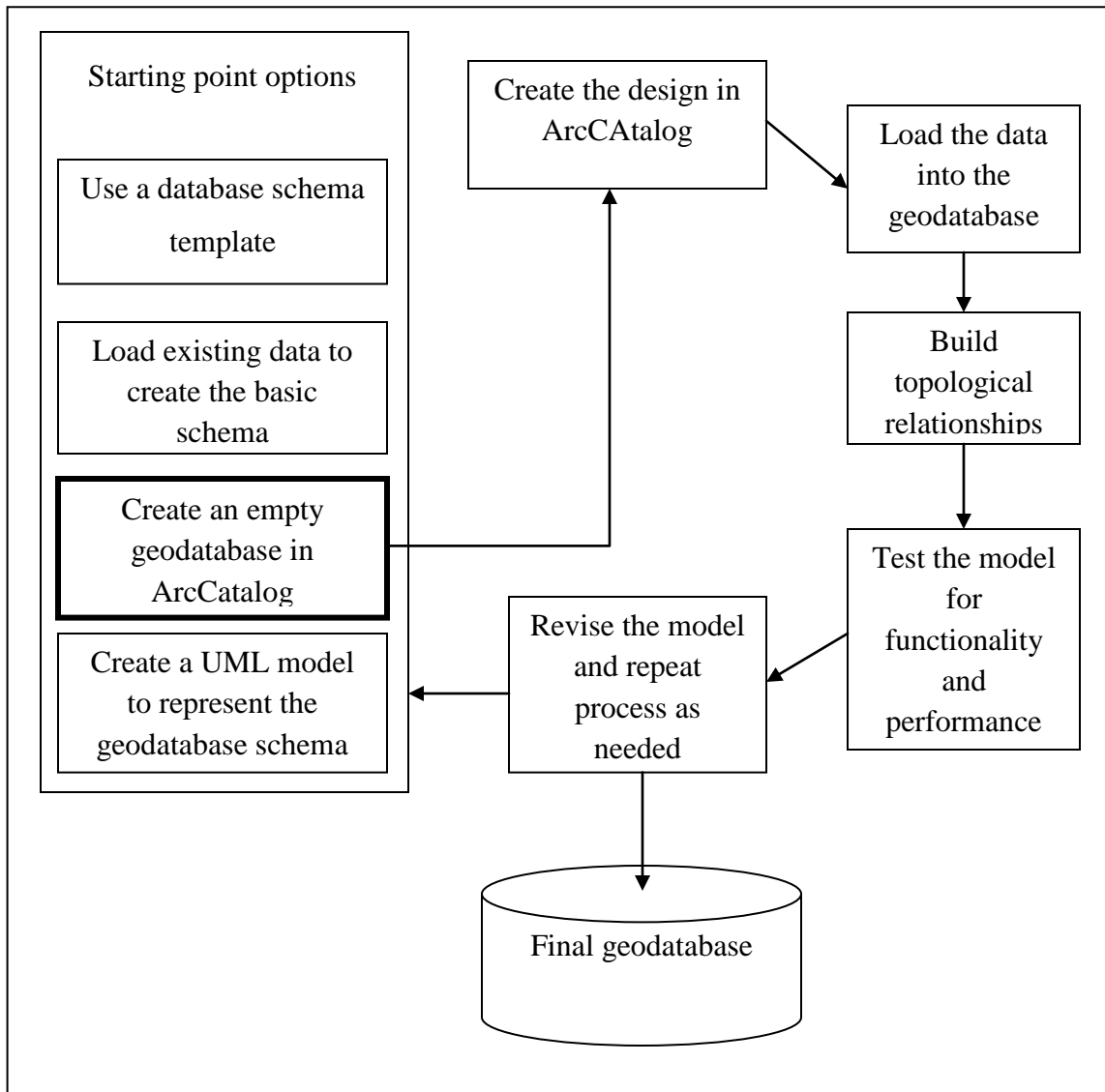


Figure 4.1: Implementation work flow for building and populating the geodatabase

4.5.1.1 Creating the Geodatabase and Loading the Data

An empty personal geodatabase was created in the ArcCatalog. The datasets were established and the empty feature classes were added to the corresponding datasets according to the structure mentioned in the conceptual design phase. In the procedure of loading the data, the best data were selected in terms of the year of collection as well as the data completeness. The best data of each feature type selected from various years to create the layers indicating the specific feature on the campus. Specifying the data to load into the empty feature classes were implemented applying two methods mentioned below:

1. *Query expression* to select the features and then load them into the corresponding empty feature class. For example the street, sidewalk, walkway, tree, hydrant, and manhole features of the various collection years were specified by SQL queries and then merged and loaded into the geodatabase. However, in some cases manual selecting implemented due to the incorrect feature coding.

Figure 4.2 shows an example of the tree points selected by query expression.

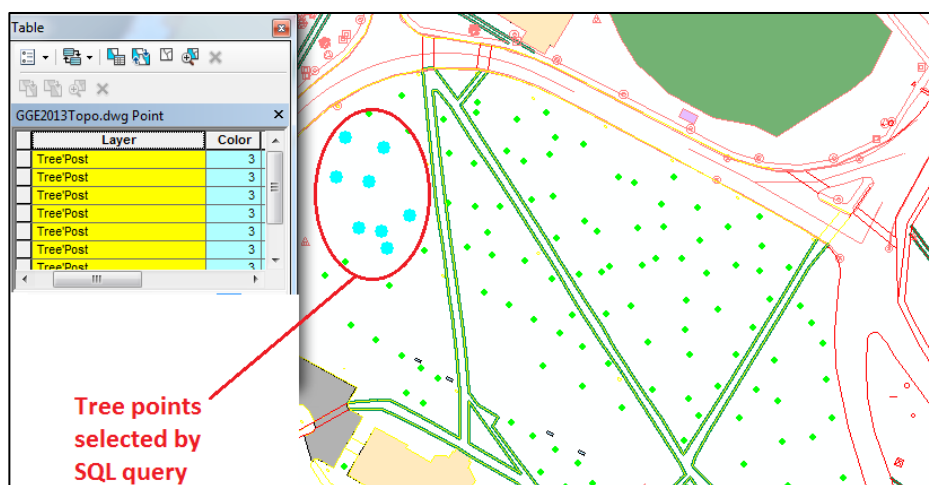


Figure 4.2: Example of tree points specified by query expression

2. **Editing session** in the ArcMap to either create the features which were not represented on the CAD maps from the other features, or to edit the existing features. Since most of the original data were provided from polylines layers on the CAD maps, the editing sessions was mostly applied to create the polygonal features. For example green area and parking lot polygons were created by the editing sessions utilizing another data types such as street, sidewalk, and walkway polylines. Figure 4.3 and 4.4 show examples of the green fields and parking lots before and after the editing.

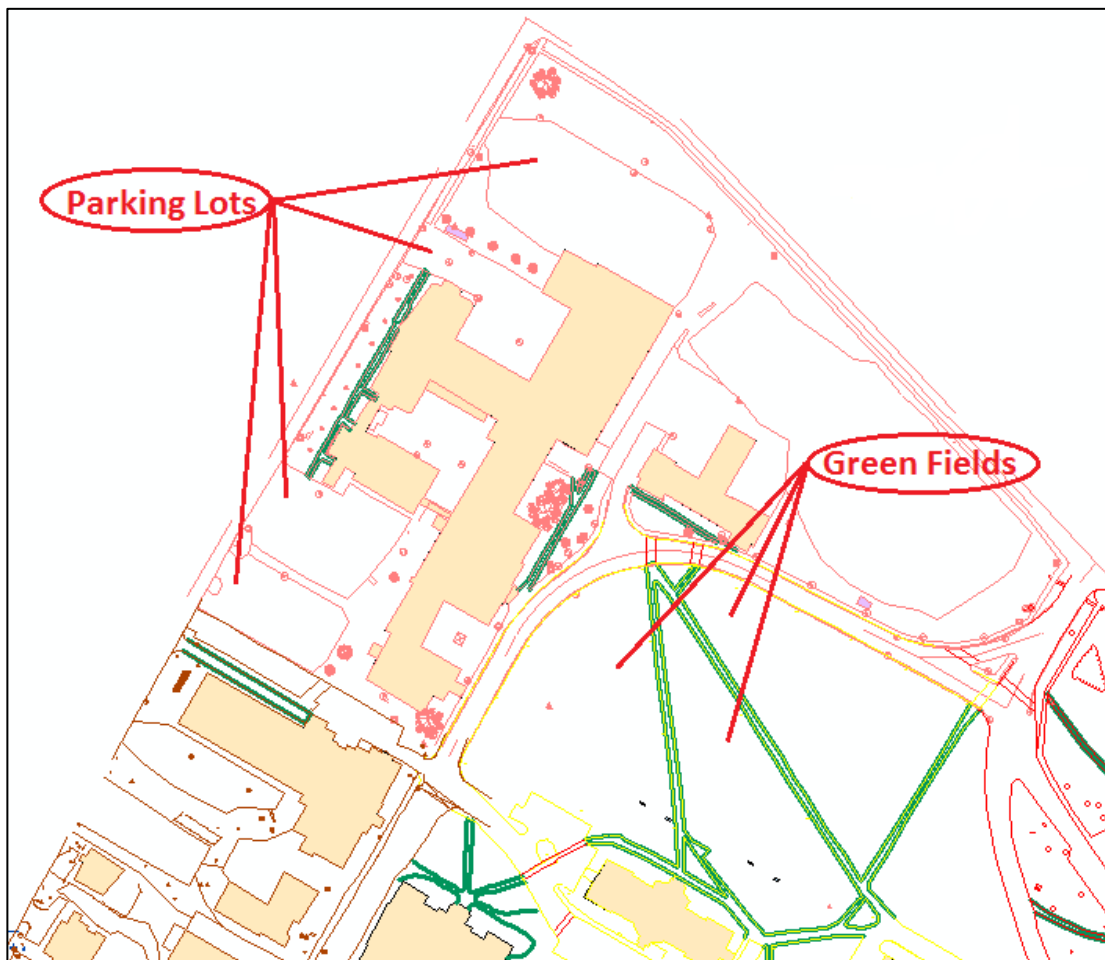


Figure 4.3: Example of parking lots and green fields before the editing

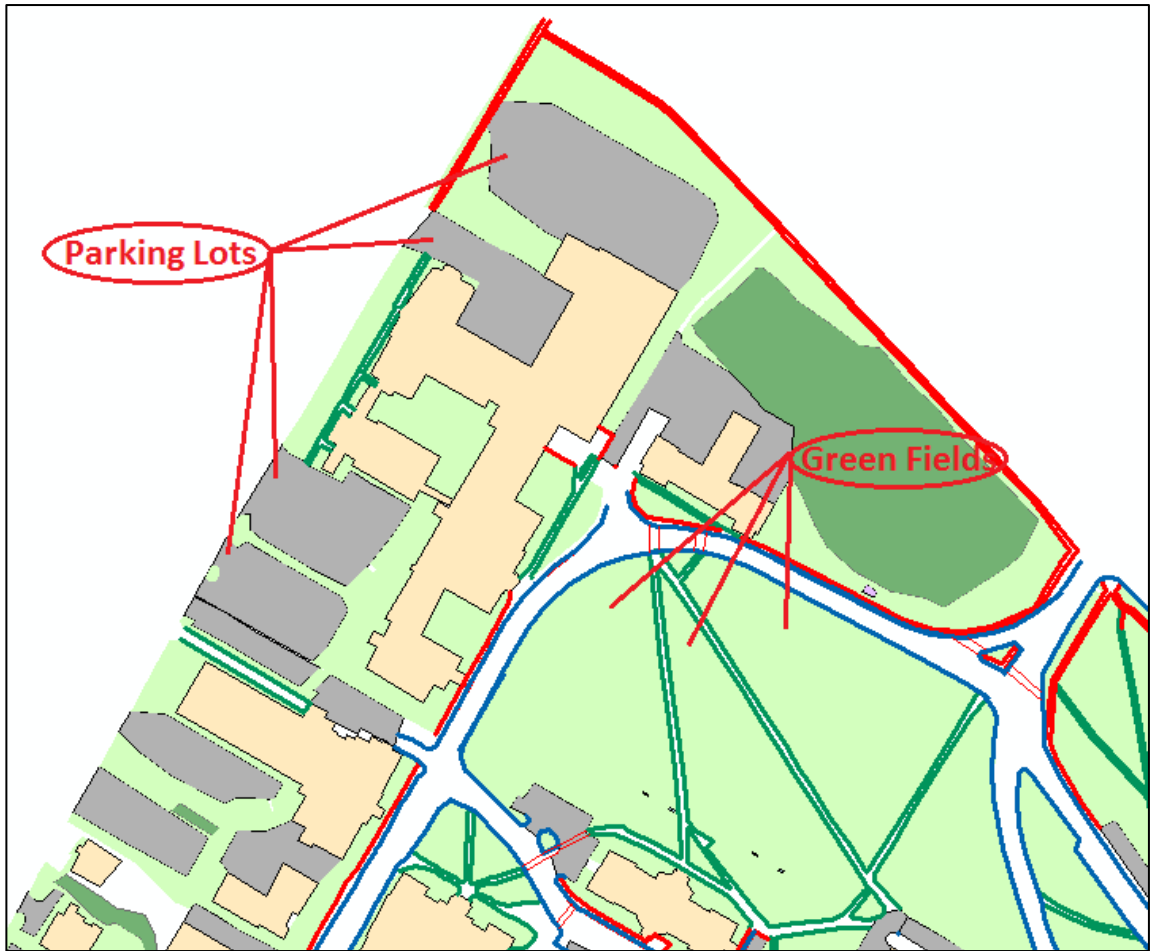


Figure 4.4: Example of parking lots and green fields after the editing

4.5.1.2 Building the Topologies and Testing the Model

The topology classes were created in each dataset and feature classes were added into the corresponding topology classes. The rules were established and applied according to the spatial relationships mentioned in the logical design phase. The topological errors were identified and then corrected for all the datasets. Figure 4.5 shows the error reports indicating there is no topological error in the datasets.

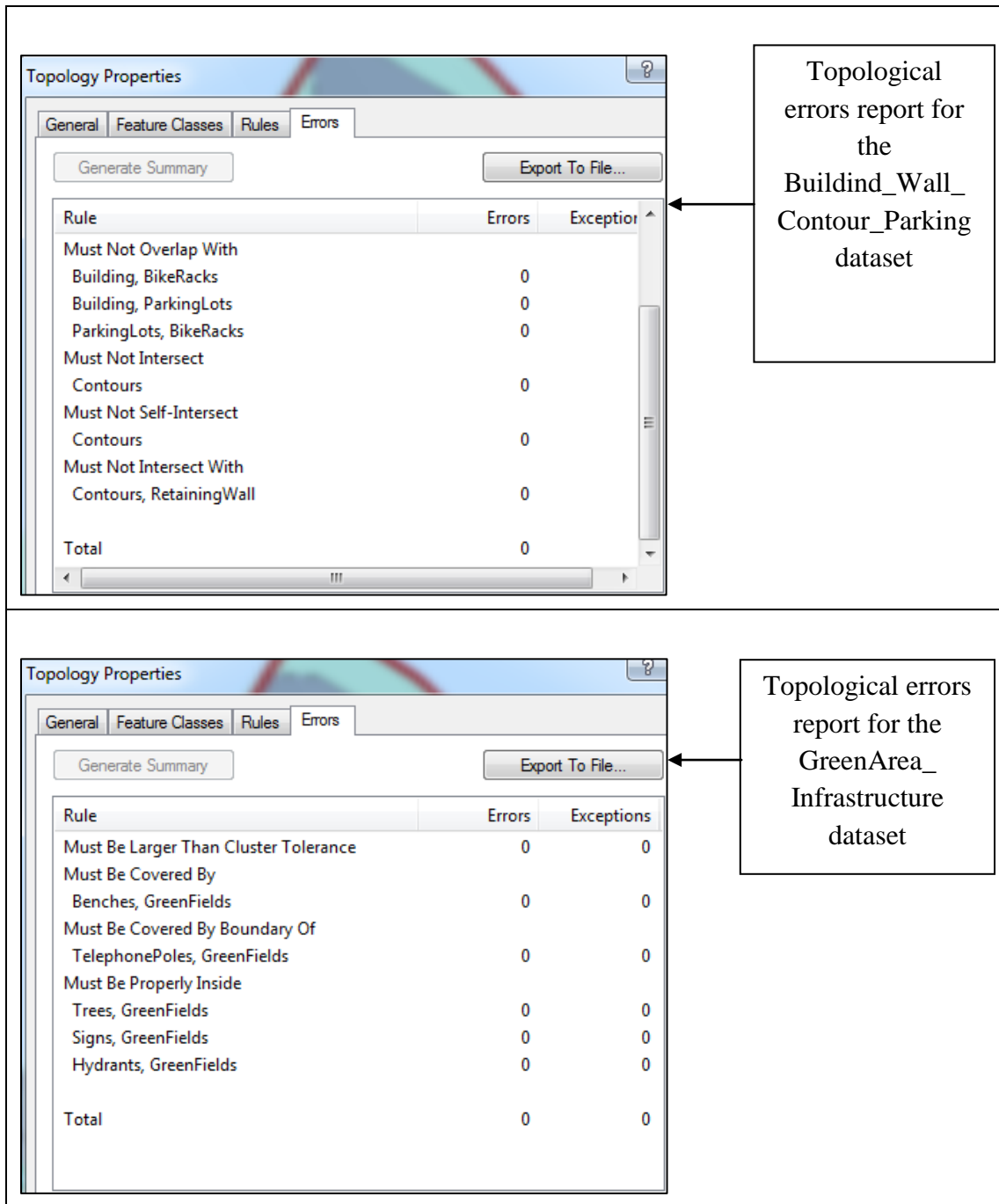


Figure 4.5: Reports of the topological errors of each dataset

Figure 4.6 shows the structure of the geodatabase created in the ArcCatalog.

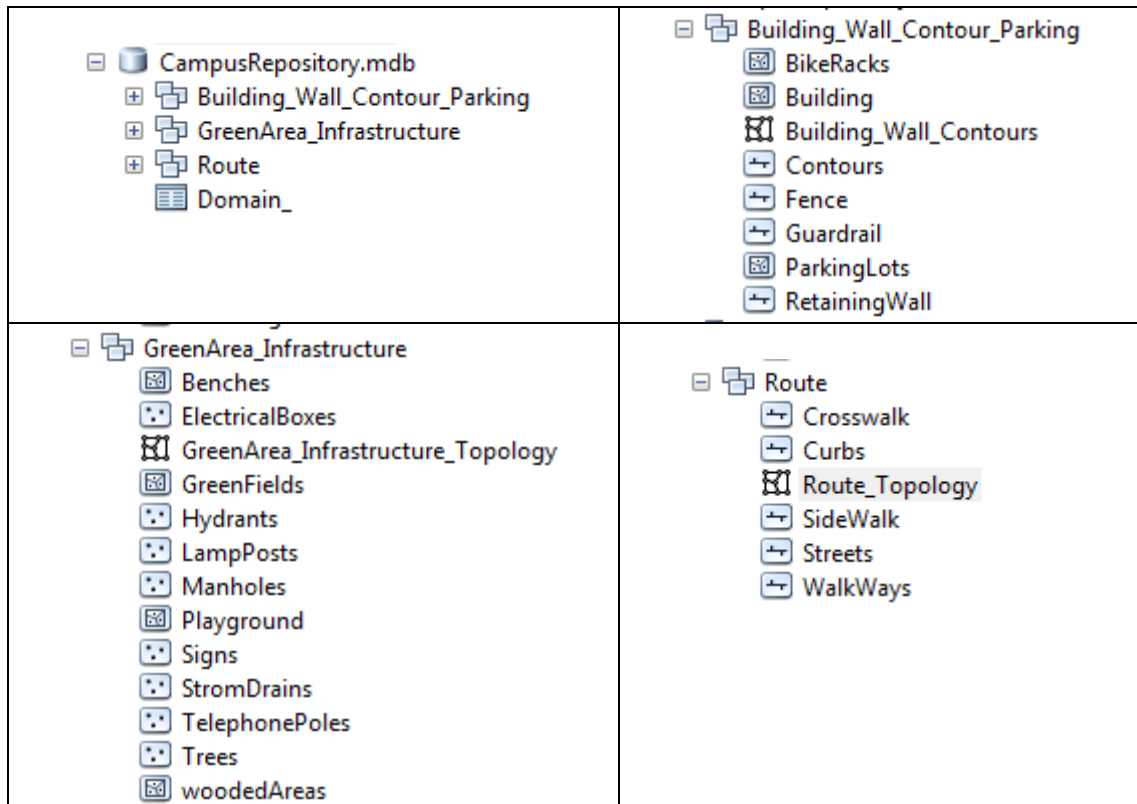


Figure 4.6: Structure of the geodatabase created in the ArcCatalog

After adding the data and establishing the topological rules, the model were tested by creating a base map of the thematic layers identified in the conceptual design along with solving the topological issues existing in the model. Figure 4.7 shows the survey camp base map including the campus’s thematic layers in a 1:6000 map scale. This scale was determined for showing the campus area in a full extent in ArcMap. Furthermore, Figure 4.8 shows an example of the campus thematic map in a 1:2500 scale which shows more details on the map (e.g. point features such as trees).

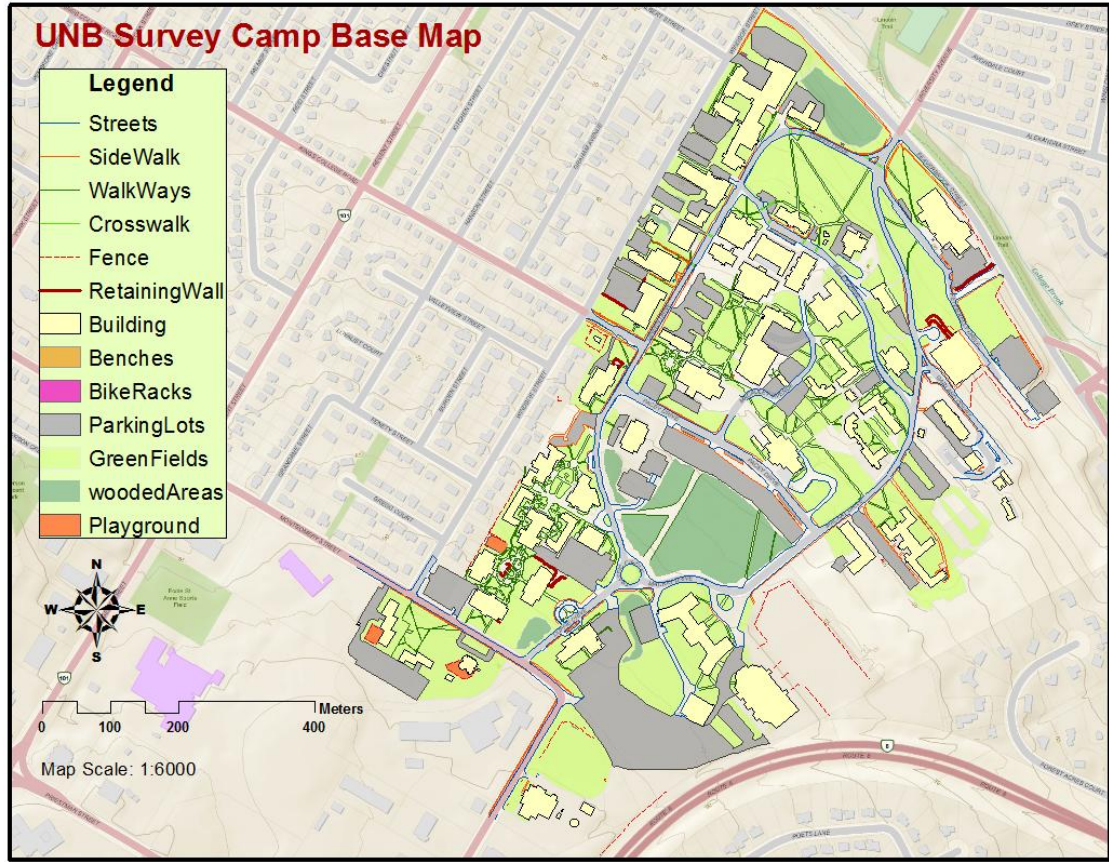


Figure 4.7: 1:6000 survey camp base map

The feature classes that represented in the base map in figure 4.7 are visible in all map scales. They are the polygonal and line features as mentioned in the map legend; whereas, the point features such as the trees and lamp posts are visible in map scale of 1:3000 and larger. Figures 4.8 shows a base map in 1:2500 scale including all the point, line, and polygon features.

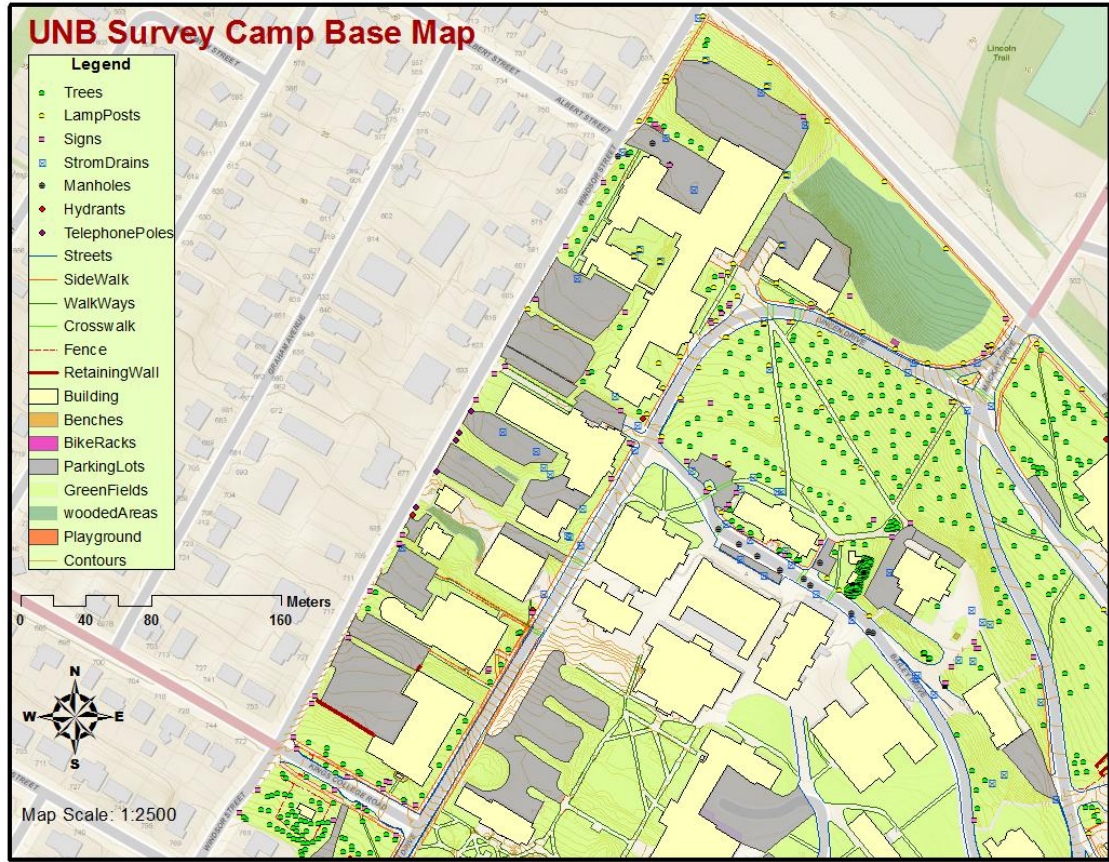


Figure 4.8: Example of 1:2500 survey camp base map

4.5.2 Design Workflows for Maintenance

There are several factors that need to be considered when developing an approach for updating/maintaining the UNB campus geodatabase with newly acquired student surveyed topographic data. Three of these are:

- How can the update process be automated;
- How does the quality of the newly acquired data (e.g. its accuracy, completeness, etc.) compare with the existing data in the geodatabase; and

- How can we ensure that changes to the topography, as represented in the newly surveyed data (e.g. new tree planted, sign removed, walkway widened, etc.), is incorporated into the geodatabase.

An analysis of these factors, such as the three mentioned above, will help in the development of an approach suitable for updating the campus database. The method that was briefly introduced in Chapter 2.3.4, Data validation and geodatabase maintenance, for maintaining the geodatabase will be described in detail in this section. This method uses GIS models to automate the maintenance process. Input data are automatically categorized and processed by GIS models and then saved as a new layer (feature class) in the associating dataset of the geodatabase. Data validation is required before applying the GIS models.

There are several difference approaches that could have been used to maintain the geodatabase and each approach has its advantages and disadvantages. Two alternative approaches are:

- Geodatabase maintenance must be performed once changes are occurred in the campus area. Changes could be mostly related to new construction areas such as new walkways, streets, or buildings. These types of changes can be easily detected while new CAD datasets are provided after surveying. That, therefore, the map dataset which indicates a newly built area must be considered for updating the old layers in the geodatabase. The updating processes are applied by editing tools in ArcMap with respect to the types of features that must be modified or created.

Since the editing tools are applied manually, based on the types and number of features, the process might be time consuming. For example, a new building could be added to the building feature class by creating a polygon feature with editing tools. Walkway lines could be edited and modified to show a new pathway.

- Once new survey camp CAD datasets indicate more appropriate data in terms of features' details and completeness. In this case, no new feature was surveyed however, the dataset might be more appropriate in comparison to the old layers. Again, manual updating processes can be applied to replace the old features with a new layer. For example, old sign features can be selected and deleted manually in the old layer and then new features are merged with the remaining features. All these processes must be performed by geodatabase administrator and manually. Furthermore, all new data must be validated in terms of tabular data as well. Therefore in all cases spatial and non spatial data must be first validated and then changes applied manually to update the geodatabase.

GIS models were used in this thesis to update the layers of each feature type in the geodatabase. The GIS models were created to fulfill the goal of maintaining the geodatabase with the new survey camp data via an automatic updating process. Automatic data categorizing, processing, and then updating are the main advantages of using the GIS models. However, automatic updating processes are applicable if a valid dataset (CAD file) be used as an input data. Therefore, the new data must be analyzed and validated prior to the maintenance processes based on the factors mentioned below:

1. *The feature codes* utilized in the *Layer* attribute field in the CAD files must be matched with the codes in the domain table. By joining the *Layer* field with the *Codes* field in the domain table, the attributes that successfully join with the codes in the domain table are identified as the valid codes. On the other hand, those which fail to joint are the features that were not correctly coded in the survey camp. Also, each feature code class must be examined visually to confirm the validity of the features related with the codes. For example street and sidewalk features could be incorrectly associated with one coded value as *ST* which indeed indicates the street features.
2. *The spatial data type* of the features must be matched with the corresponding feature classes in the datasets. Each feature type must be specified and then imported as a single feature class to the datasets. Also, some features must be converted to another data type. For example building and parking lot were defined in the framework characteristics as enclosed features; thus, these features must be converted to polygons data type and then use in the maintaining process.

After validating and importing the new feature classes into the datasets, the GIS model can be used to replace the old features with the new features. In the other words, the GIS model updates the layers using the new survey camp data. In fact, the old data are analysed based on the common spatial relationship with the corresponding new data. Finally the old data are specified and then replaced with the new data. Since the various data types including points, lines, and polygons stored in the repository, each data type is individually analysed according to the specific spatial relationship exist between the old and new data.

Figure 3.5.4 illustrates the procedure of validating and maintaining the geodatabase.

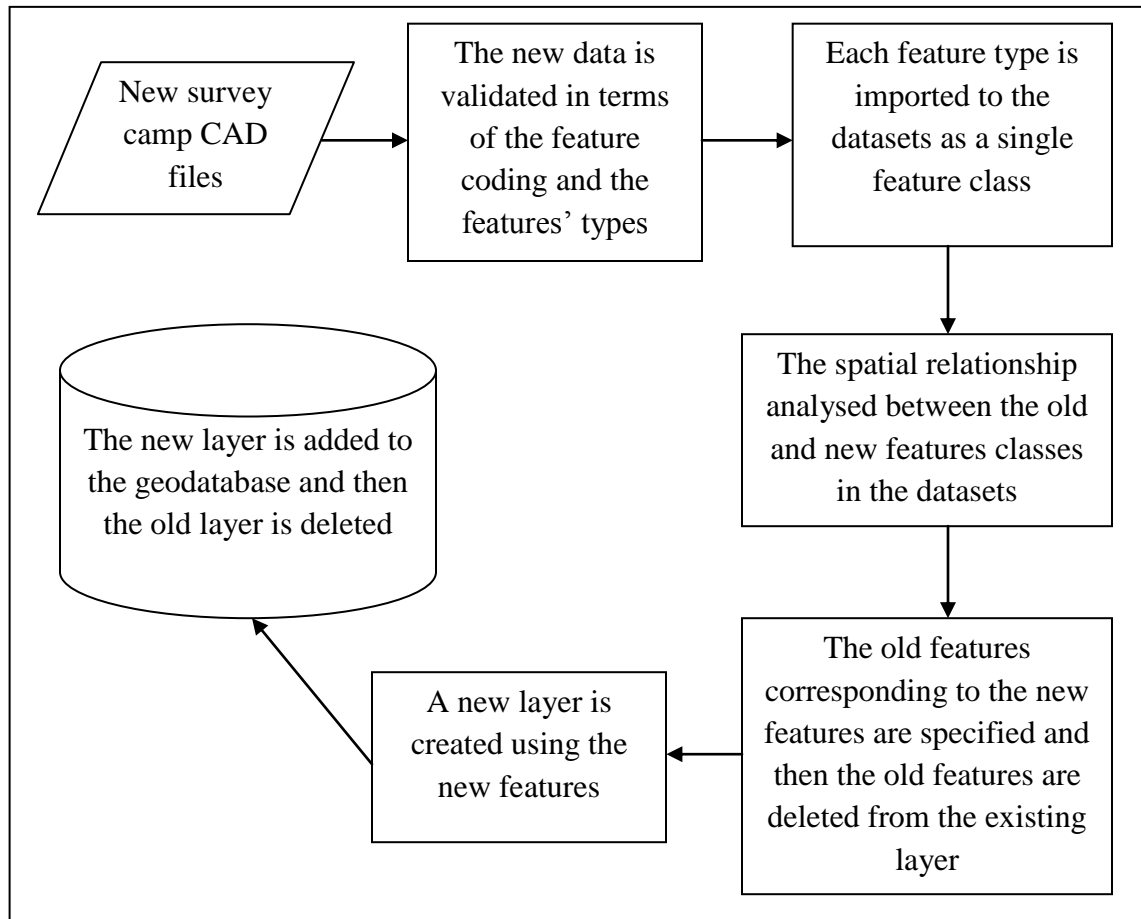


Figure 4.9: Geodatabase maintenance workflow

4.5.2.1 GIS model for Geodatabase Maintenance

Various geoprocessing tools were included in the GIS model work together to automatically update the layers in the geodatabase. If the input data is valid in terms of the attribute coding and data type, the result of the GIS model will be a feature class containing the data updated in the geodatabase. The processes mentioned below are performed by the GIS model.

1. ***Creating feature classes*** from all the different feature types in the new survey camp CAD files which utilized as the input data in the GIS model.
2. ***Adding a new attribute field*** to the new feature class that is used in the updating process. After a new feature class was imported into the dataset, a new attribute field must be added to the attribute table and then the field must be populated with a value. In this case, the “old” text value was selected to populate the field.
3. ***Spatially joining*** the new and old data to automatically specify the old features which must be replaced with the new features. However, different type of spatial joins was utilized with respect to the various data types. For example building features must be joined based on the spatial intersection occurred between the old and new building polygons. On the other hand, manhole features must be analyzed according to the closeness occurred between the new and old manhole points. In fact, in this step, the “old” value added in the first step is associated with the old features which are spatially joined with the new features. Thus, the old features can be automatically specified and then be used in the next step.
4. ***Extracting*** the features which were not specified as “old” in the spatial join process. By using a SQL query, the features associated with “old” value are eliminated and then a feature class produced including the remaining features.
5. ***Merging*** the new features with the features resulted from extracting process. After the merging, a new feature class is produced and stored in the dataset and then will be used on the campus map.

On the next page, figure 4.10 illustrates the tools and procedure performed in the GIS model along with the parameters, aspects, and the outcomes of each tool. As it is shown in the figure, once the input data are selected (point and polylines layers), the process starts with categorizing the data into different types of feature classes.

After categorizing, data are processed for matching the old and new data. Once corresponding old and new data are determined, with spatial join geoprocessing tool, actual updating process is performed. The updated layer is a feature class consists of the newly surveyed areas and the old surrounding areas.

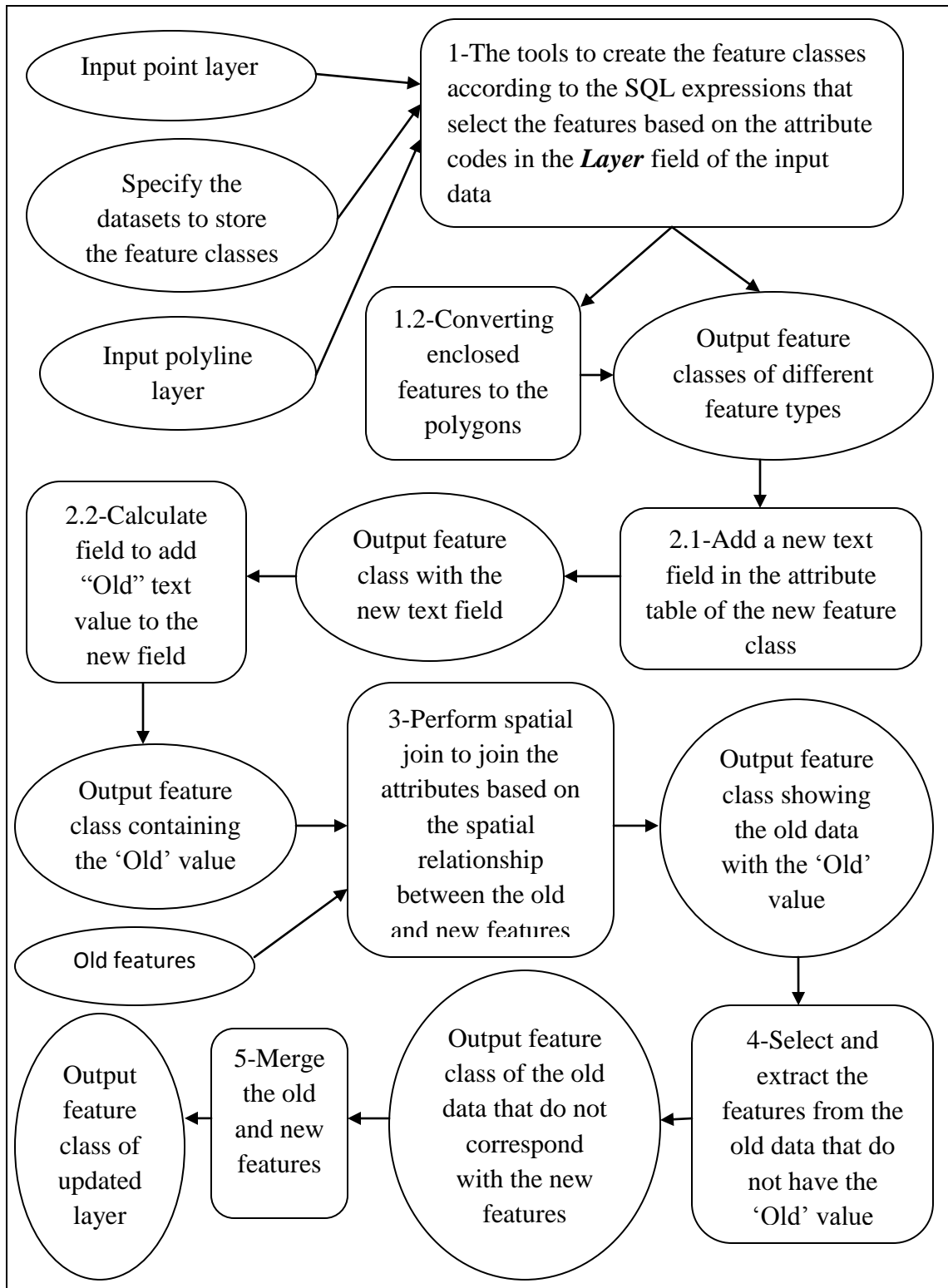


Figure 4.10: Shows the tools and processes applied in the GIS models


4.5.3 Geodatabase Documentation

There are important elements mentioned below to consider while documenting the geodatabase.

(<http://webhelp.esri.com/arcgisserver/9.3/java/index.htm#geodatabases/docume-1980433582.htm>)

1. **Datasets:** To show the feature classes in the dataset along with the topology classes established for them.
2. **Domains:** Represents the list or range of valid values for attribute columns. These rules control how the system maintains data integrity in certain attribute columns.
3. **Spatial relationships and spatial rules:** Represents the topological rules established between the feature classes in each datasets.
4. **Map layers:** The spatial representation of the map layers must be mentioned in terms of the data content and symbolization.

Because all the elements mentioned above have been already described with details in the steps of geodatabase design and implementation; therefore, figure 4.11 provided to give an example of the **Route** dataset documentation to show the spatial and tabular details of the feature classes in it.

 **Route Dataset**

Simple feature class							Geometry	Polyline
Streets							Contains M values	No
							Contains Z values	No
Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length	
OBJECTID	Object ID							
SHAPE	Geometry	Yes						
SHAPE_Length	Double	Yes			0	0		
FeatureCode	String	Yes					2	
FeatureType	String	Yes		Codes			50	

Simple feature class							Geometry	Polyline
WalkWays							Contains M values	No
							Contains Z values	No
Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length	
OBJECTID	Object ID							
SHAPE	Geometry	Yes						
SHAPE_Length	Double	Yes			0	0		
FeatureCode	String	Yes					2	
FeatureType	String	Yes		Codes			50	

Simple feature class							Geometry	Polyline
Crosswalk							Contains M values	No
							Contains Z values	No
Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length	
OBJECTID	Object ID							
SHAPE	Geometry	Yes						
SHAPE_Length	Double	Yes			0	0		
FeatureCode	String	Yes					2	
FeatureType	String	Yes		Codes			50	

Simple feature class							Geometry	Polyline
SideWalk							Contains M values	No
							Contains Z values	No
Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length	
OBJECTID	Object ID							
SHAPE	Geometry	Yes						
SHAPE_Length	Double	Yes			0	0		
FeatureCode	String	Yes					2	
FeatureType	String	Yes		Codes			50	

Simple feature class							Geometry	Polyline
Curbs							Contains M values	No
							Contains Z values	No
Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length	
OBJECTID	Object ID							
SHAPE	Geometry	Yes						
SHAPE_Length	Double	Yes			0	0		
FeatureCode	String	Yes					2	
FeatureType	String	Yes		Codes			50	

Figure 4.11: Example of dataset documentation

4.6 Lesson Learned

In designing a geodatabase model, it is critical to initially examine and interpret the source, usage, and type of the data. These factors allow having a better and appropriate understanding from the design goals. By understanding the goals, a suitable model will be built by which useful management and organization of geospatial data is possible. A well designed geodatabase model significantly improves the quality of outcomes and products (e.g. the thematic and topographic maps showing the data content and specifications).

It was found from designing the geodatabase that it is very important to consider and define the spatial relationships between the objects. These kinds of relationships allow improving the system reliability in terms of categorizing, interpreting, and analysing the geospatial data. For example, when a new set of point features are added to the repository; then, it is possible to examine if the point features have been correctly located and specified or not (i.e. the hydrant and lamppost points must be inside the green fields). Furthermore, providing the coded value domains allowed establishing a particular validating system for the tabular information by which system could help to satisfy the goal of validating the input data and respectively identifying the map data easier and faster.

Chapter 5 Web-GIS Service Development

5.1 Introduction

In this chapter the development of the Web-GIS application will be described from visualizing and publishing the survey camp topographic map layers and tabular contents over the web. Section 5.2 will first introduce the techniques applied in the developing of the Web-GIS service. Then, further information and illustrations will be given in section 5.3 regarding the structure and functionality of this Web-GIS service.

The Web-GIS performance regarding the data visualization and representation of the spatial and non-spatial characteristics of the survey camp information model will be examined in section 5.4.

5.2 Web-GIS System

The web system has been set up based on ArcGIS server in the role of GIS server and ArcGIS JavaScript API in the role of the web programming framework. The input data for the web service is provided from the previous steps including the UNB campus' map layers that were created from survey camp topographic data. The base map has been published as an interactive map service. The map service provides a series of functionalities in respect to the spatial framework design and characteristics. Functionalities are served by the ArcGIS server and ArcGIS JavaScript API respectively based on ArcGIS REST API specifications.

ArcGIS Server REST API stands for Representational State Transfer. REST API provides a Web interface to services that hosted by ArcGIS Server. All GIS data and services which are published by ArcGIS server are accessible by the REST API through a hierarchy of endpoints or Uniform Resource Locators (URLs). Each published service has a unique URL. The URLs are the end points to the published services such as maps, geocode, geoprocessing, geometry, and image services. ArcGIS server has been integrated with Windows Internet Information Server (IIS). IIS is the web server software in Microsoft Windows. There are other available web servers such as Apache or Apache-Tomcat which are popular and are being used by several Web-GIS applications. However, ArcGIS server is set up with IIS by default during the installation process. Figure 5.1 illustrates the main structure of the Web-GIS system according to the utilizing technologies and tools.

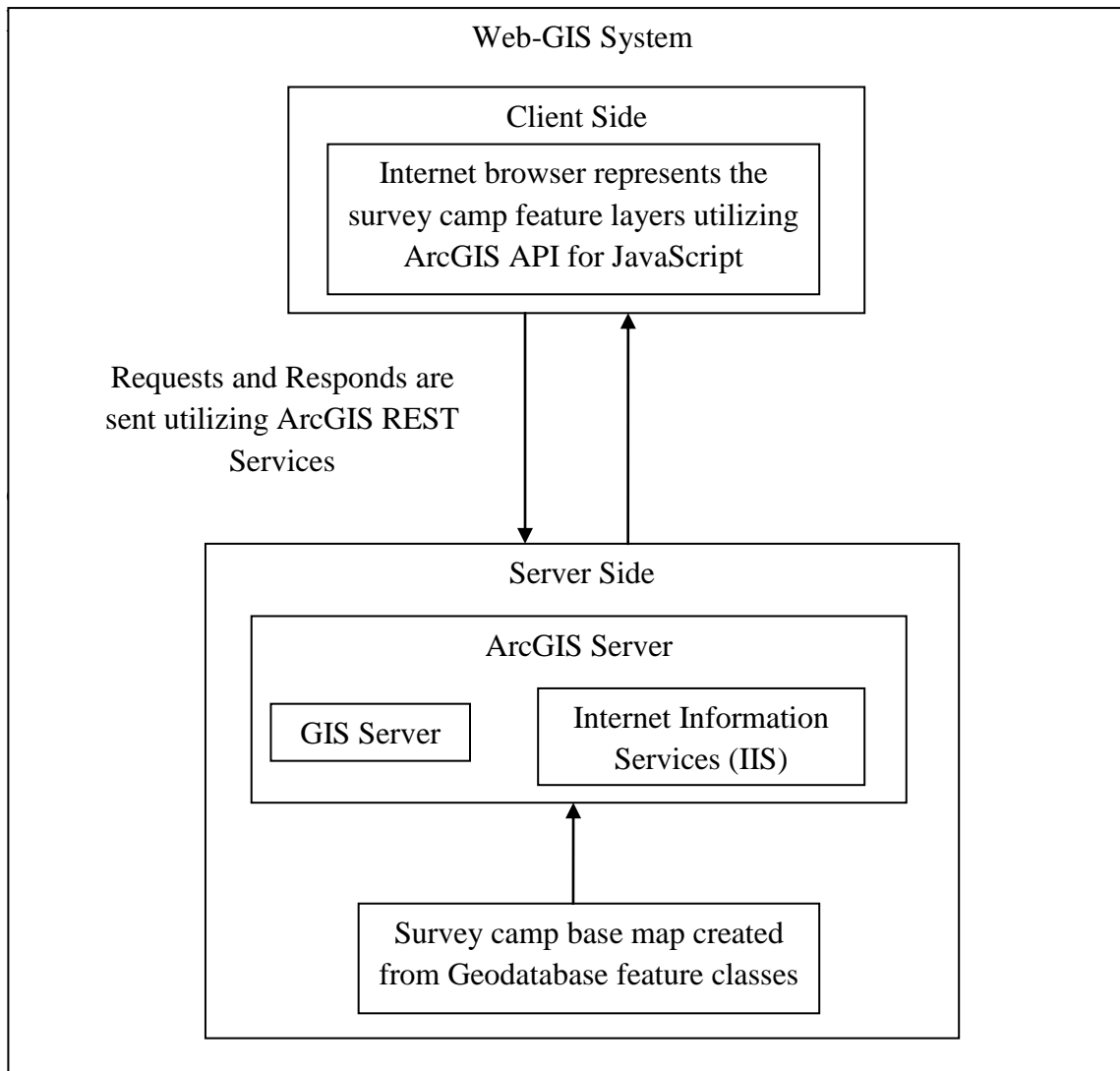


Figure 5.1: General illustration of the Web-GIS system and architecture

5.2.1 Server Side

ArcGIS server was used in the service development as the GIS server to perform the main process of managing the server side system. ArcGIS server is designed to provide a wide range of GIS functionality in a server machine (e.g. the computer system which contains the map data and datasets) to the client side (e.g. computer and mobile internet browsers) through the web. ArcGIS server helps to have a strong managing framework for developing Web-GIS systems that can be utilized in browser-based applications.

ArcGIS server is administrated and set up by using either ArcCatalog or Manager to publish and administer the GIS web-based services. Manager is a Web application and a browser-based administration system which helps to provide Web-GIS services. On the other hand, ArcCatalog can be used as a desktop application to provide and manage map services publishing by the ArcGIS server.

Each one of the GIS services has a unique URL pointing to a directory on the server machine. The URLs will be used in the JavaScript code to retrieve the map services spatial and non-spatial contents and then visualize the map layers at the client side through the internet browser. According to the service type and data hierarchy the URLs are different. Further illustration will be given for the utilized services and their URLs. Figure 5.2 shows the ArcGIS server system architecture in terms of the server side and client side administration components and tools as well as the system hierarchy.

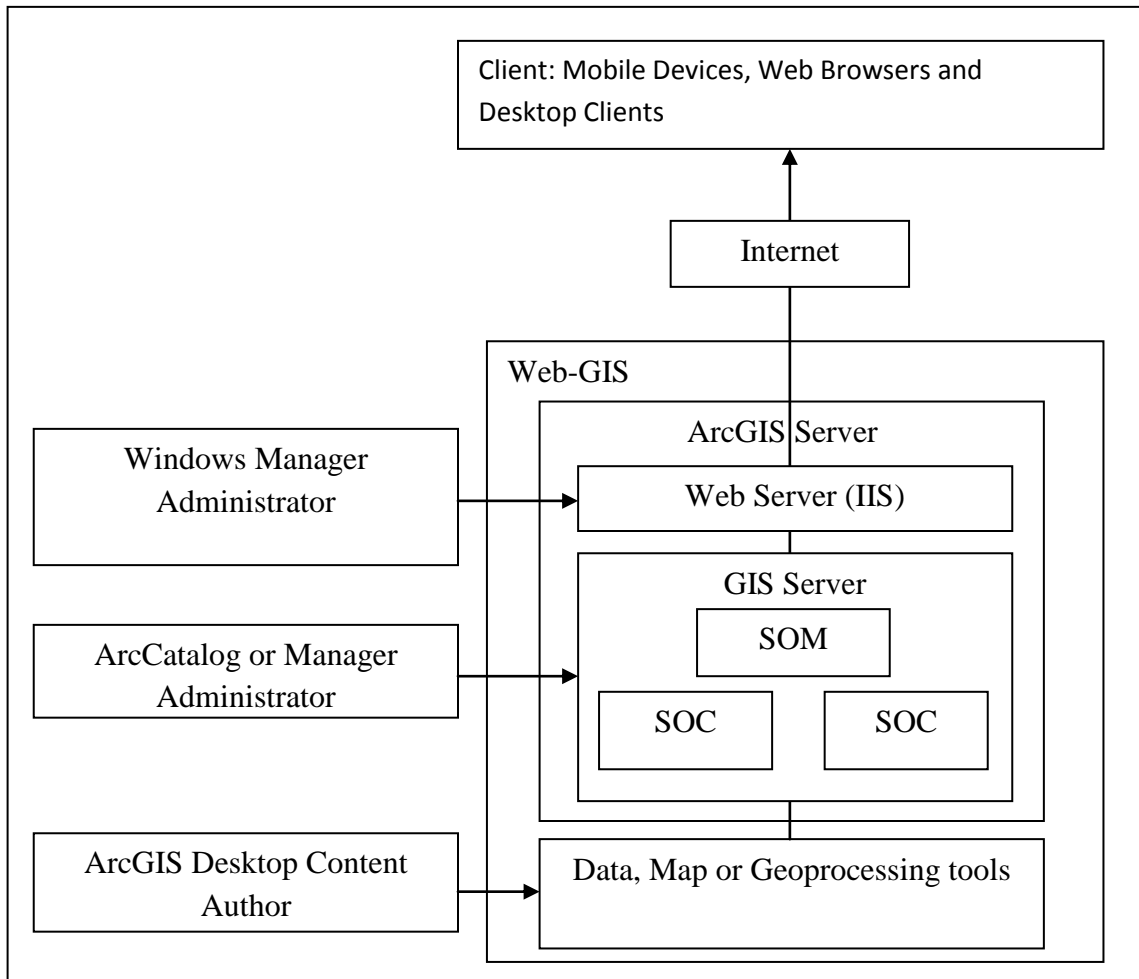


Figure 5.2: ArcGIS server system architecture

5.2.2 Client Side

The most important contribution in developing the Web GIS service in this thesis was in programming the client side JavaScript code and producing the applications. As mentioned in the section 5.2, the ArcGIS JavaScript API was utilized for the web service development. ArcGIS JavaScript API is a programming framework which provides a wide range of functionalities and capabilities in terms of map visualization and GIS tasks implementation (e.g. visualizing feature layers, and performing query tasks). Figure 5.3 illustrates the client side system general structure regarding programming languages and technologies.

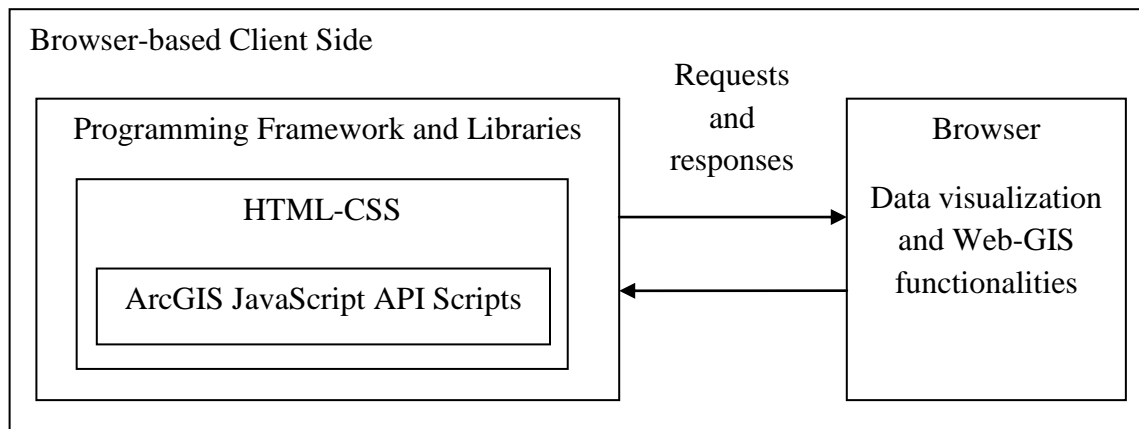


Figure 5.3: Shows the programming framework and libraries

5.3 Web-GIS Programming structure and Properties

In this section, the Web GIS development procedure will be discussed in terms of the map data, service functionalities, and code properties. As a reason, web service code will be divided into four key sections and each section will be described in terms of technical properties, specifications and elements. Thereby, the Web GIS service will be assessed according to its efficiency and usability of publishing and representing the geospatial reference framework characteristics.

There are four critical parameters mentioned bellow. These parameters are the main factors to consider in programming and development of the main body to structure the JavaScript and HTML code. The critical parameters are to:

- 1- Specify the input data and data types.
- 2- Define the main web mapping applications and functions.
- 3- Define the methods for calling the functions and presenting their results.
- 4- Specify the output data and data types.

Figure 5.4 illustrates the general structure of the code programming. Figure 5.4 describes the overall procedure in developing the Web-GIS code structure. The service performance and more details for the parameters mentioned above have been given in the figure as well.

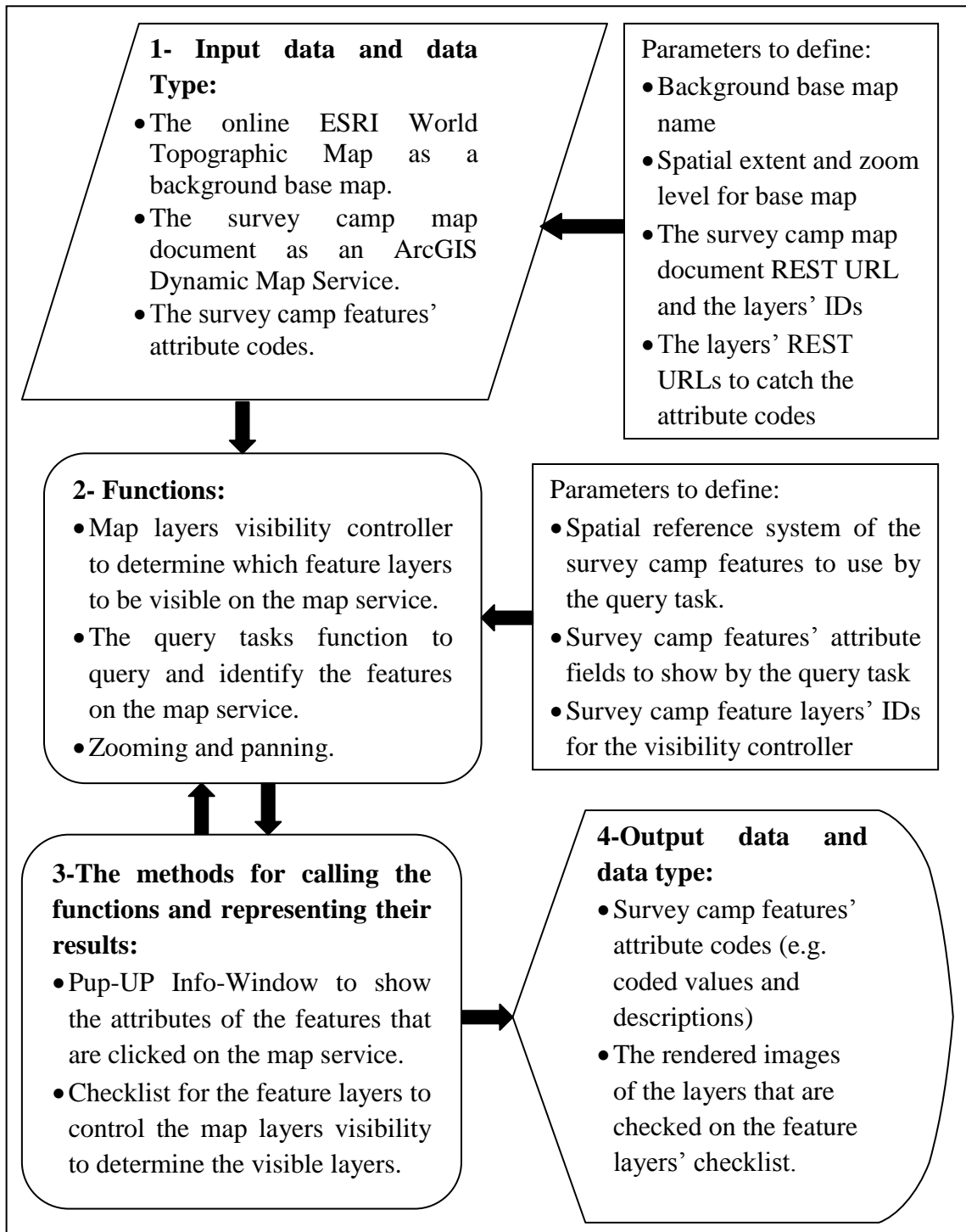


Figure 5.4: General aspects of the Web-GIS service code and performance

5.3.1 Input Data and Data Type

As illustrated in the figure 5.4, there are three various types of input data which have been used as the Web-GIS map content. First type of input data is the ESRI World Topographic Map, which is used as a background base map in the application. ESRI World Topographic Map is obtained from ArcGIS online map service and there are parameters to set regarding the spatial extension and zooming level. These parameters allow the map service to start while showing a specific geographical area of the background base map (e.g. UNB campus area) according to the input values that were specified in the code as the spatial extension (e.g. the spatial extension of the survey cam layers) and zooming level (e.g. a zooming level that shows the campus area appropriately). The utilized values can be found in the code mentioned in Appendix II.

The second type and the main spatial input data are the survey camp layers. The layers are overlapped with the background base map while displaying on the map service. This ability of overlapping with background base map allows the users to have a better interpretation from the feature layers. The Survey camp data is added to the map service using ArcGIS Dynamic Map Service Layer JavaScript Class and via a REST URL. The REST URL points to the map document (.MXD file) directory on the server machine. While the data being added, each feature layer is identified with a specific layer ID number. The ID numbers, are specified by Image Parameters JavaScript Class, and are usable for processing each layer separately in the functions. For example, the layer visibility controller function controls the visibility of layers based on their IDs.

The main reason for using the ArcGIS Dynamic Map Service Layer Class is its capability of creating images for each individual map layer, while using only a single REST URL pointing to a specific map document. Moreover, this JavaScript Class can dynamically render new images from the feature layers each time users pan or zoom over the map service. This ability made the procedure of code programming easier in comparison with using of WMS Layer or Tiled Map Service Layer JavaScript classes.

The third type of input data is the tabular data that are known as feature attributes. The attribute data were added to the map service by using QueryTask JavaScript Class and via a set of REST URLs. In fact, for the attribute data, more than one single REST URL was needed, and each URL must point specifically to an individual feature layer directory. Therefore, based on the number of survey camp feature layers that included in the map document, different REST URLs were utilized in the QueryTask JavaScript Class.

There are also a set of parameters for the attribute data to set by using a JavaScript class which is named as Query. These parameters define various factors of the feature attribute data such as the attribute fields, feature geometry, and spatial reference system. After setting the parameters mentioned above, the QueryTask Class will be able to retrieve the attributes of each feature that selected on the map service. To illustrate the programming procedure of adding the input data into the map service, table 5.1 shows the utilizing ArcGIS JavaScript API classes and their properties.

Table 5.1: ArcGIS JavaScript API classes and properties utilized for input data

Input Data	ArcGIS JavaScript API Class	Properties and Parameters
ESRI Word Topographic Map	<ul style="list-style-type: none"> • esri.Map • esri.geometry.Extent 	<ul style="list-style-type: none"> • Base map name • Spatial extension • Zooming level
Survey camp feature layers	<ul style="list-style-type: none"> • esri.layers.ArcGISDynamicMapServiceLayer • esri.layers.ImageParameters 	<ul style="list-style-type: none"> • REST URL • Feature layers ID numbers
Survey camp feature attributes	<ul style="list-style-type: none"> • esri.tasks.QueryTask • esri.tasks.Query 	<ul style="list-style-type: none"> • REST URLs • Attributes fields of the features • Feature geometry • Spatial reference system

5.3.2 Functions

The most important part in developing the map service was to provide a set of useful functions. Map service capability is depended on its functionality in terms of data visualizing and data retrieving. There are three sorts of functionality that have been provided for the map service.

Two types of are to control the data visualization including the feature layers visibility controller along with the tools of zooming and panning. The visibility controller function allows users to select which feature layers to be displayed on the map service. Visibility controller works based on the map layers' IDs defined as the input data parameters that described in the section 5.3.1 . Table 5.2 provides a list of JavaScript classes and describes their performance in the visibility controller function.

Table 5.2: List of the ArcGIS JavaScript API classes for the visibility controller

ArcGIS JavaScript API Class	Functionality	Parameters
dojo.query	Returns the HTML inputs that contain the feature layer IDs to catch the ID numbers	HTML inputs class
for-if conditional loop function	Checks if any of HTML inputs checked on the map service	
visible.push	Add the layer ID number to the visible variable	Inputs ID
setVisibleLayers	Takes the layers' IDs and set them to be visible	Visible variable that contains numeric values

Additional to the JavaScript classes and their performance listed above, figure 5.5 shows a diagram illustrating how the classes work together in a loop in order to control the layers visibility on the map service.

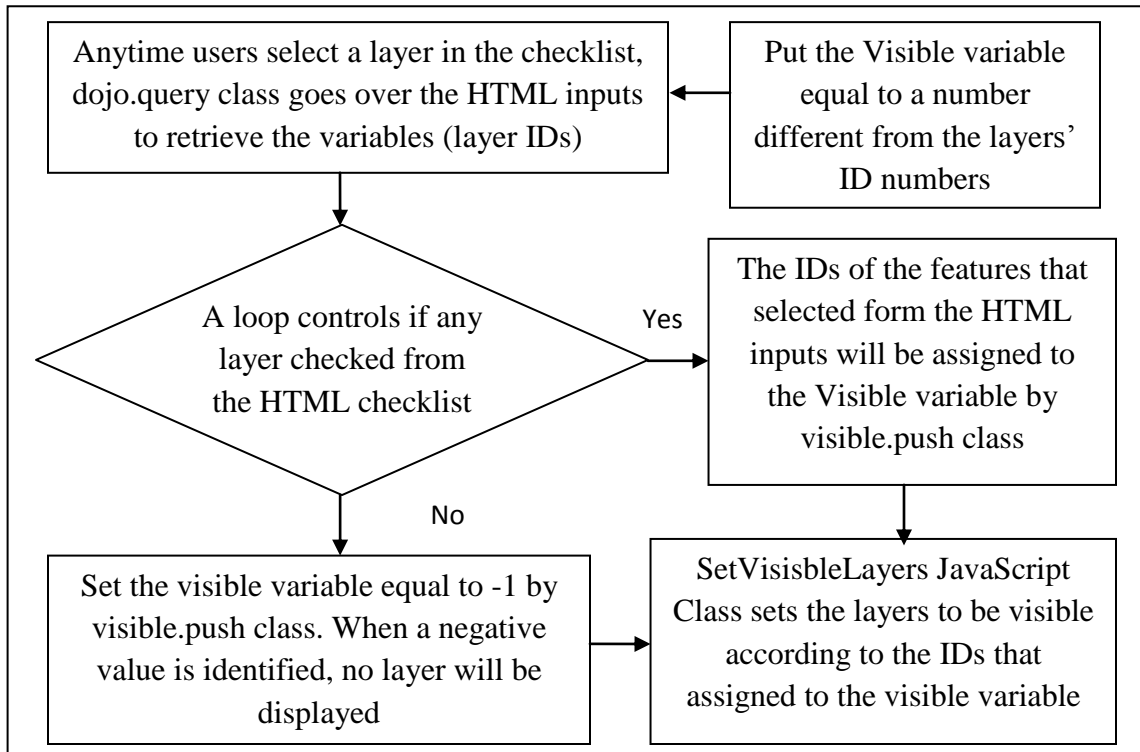


Figure 5.5: Loop function of the visibility controller of the map service

As mentioned above, the visibility function is a combination of various JavaScript classes. This function is connected to the ArcGIS Dynamic Map Service Layer Class, which performs the main process of displaying the map layers, via the SetVisibleLayer Class. Once a map document is added to the service, the displaying status of map layers is set to visible by default; in fact, all the layers' ID numbers of a map document are determined as visible by default. However, once users select a set of layers via the HTML inputs, the visibility will be limited only to the layers that their IDs have been set to visible by SetVisibleLayer Class. Therefore, the ArcGIS Dynamic Map Service Layer Class will only display the layers that their ID numbers have been included to the visible variable of the SetVisibleLayer Class.

The third type of functionalities, the Query Task Execution function, allows the users to identify the features and retrieve their attribute data. The Query Task Execution function calls the other JavaScript classes from the other sections of code, which will be described in section 5.3.3 including the methods of calling the functions. This function executes the actual querying process for retrieving the attributes via running a set of JavaScript classes. This function calls all other sections including the QueryTask classes by which input attribute data have been defined via the URLs, and the function of result showing as well. In other words, the Query Task Execution function connects various parts of the code from input attribute data to the final pup-up window and finally results in showing the attributes of the selected feature. Table 5.3 shows the JavaScript classes utilized in the function and describes their performance.

Table 5.3: The JavaScript classes utilized in the Query Task Execution function

ArcGIS JavaScript API Classes	Performance Description
map.infoWindow.hide	Hides any pup up window remained opened from previous selection
map.graphics.clear	Clears any selected features remained from previous selection
featureSet = null	Makes the feature variable array empty from any selected features
queryTask.execute	Executes the QueryTask class to retrieve the attribute data from input URLs and according to the defined query parameters with the Query class.

Each time users select a feature, the query execution function executes the mentioned JavaScript classes in above. For better understanding of the function performance, the figure 5.6 shows a diagram illustrating the function and describes its performance procedure.

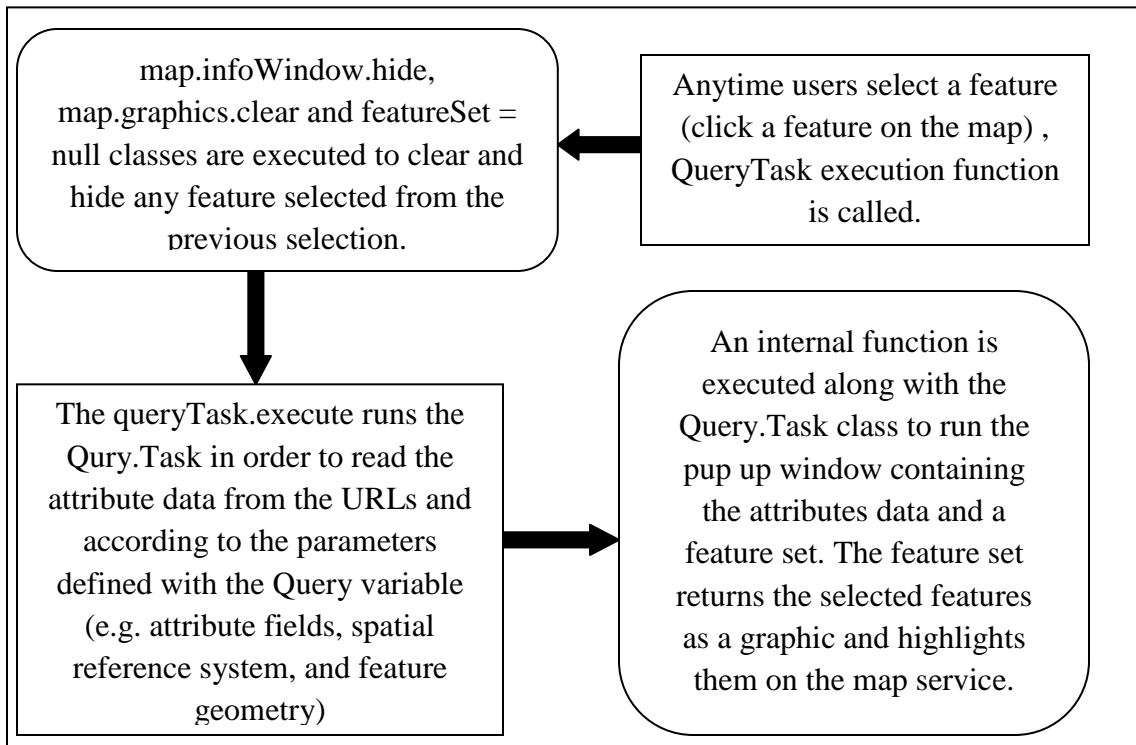


Figure 5.6: Work of the QueryTask execution function

5.3.3 Calling the Functions

There are two major methods for calling the layer visibility controller and the feature attribute query functions. Each of these functions can be easily called in the map service when users simply check the layers in a checklist and by clicking on the features on the map service. The methods explained below along with the illustrations.

1. *The feature layer checklist* is connected to the visibility controller function via HTML inputs and calls it each time users check one of the map layers. With respect to the map service capability of rendering images for each of feature layers, creating a checklist for the map layers determined as the best method to control the map layers in terms of visualization and displaying. HTML inputs utilized to construct the checkboxes indicate the map layers on the map service. There are parameters to set in the HTML codes for specifying the layers' IDs in each HTML input.

The visibility function is called each time a user click on a checkbox. Consequently, the layer ID associating with the HTML input will be obtained and then added to the visible variable in the visibility controller function. The REST service helps to identify the layers' IDs in order to write the HTML codes which construct the checkboxes. Figure 5.7 shows the REST service containing the list of map's feature layers and their associating IDs.



Figure 5.7: Layers' IDs in the REST service API

2. *The Onclick method* to call the attribute query function (Quer Task Execution function). With the Onclick method, each time users click on a feature the query task function is called and the whole process of querying function is executed. After clicking on the features the attributes of the features clicked on the map are represented in a pup-up info-window.

5.3.4 Output data and the data type

There are two types of output data published and represented by the map service including the survey camp layers as well as the features' attributes data.

1. *The survey camp feature layers* are represented on top of the ESRI world topographic base map in order to illustrate the framework specifications in terms of introducing the key features on the campus. The layers correspond with the feature classes of the geodatabase that built in the second phase of this project. However, all the features represented on the map service are polygons. In fact, another geodatabase created to store the polygonal feature layers of the map service.
2. *The features' attributes* which indicate the feature coding system, feature types, and the features' data type. The attributes are retrieved by clicking on the map; however, the point and line features are not detected on clicks. In order to solve the problem, all the features were converted to polygons by which the server could detect the features on clicks and then retrieve the associating attributes and geometry. In spite of representing all the features with polygons, the features' data type attributes specified according to the data type defined in the geodatabase logical design phase. For example, the street features are shown as polygons on the map service; although, the data type is shown as line indicating the street features were stored as lines in the main geodatabase. It allows fulfilling the goal of representing the framework specifications with the map service. Table 5.4 Shows the map layers along with the associating attributes.

Table 5.4: List of the layers and attributes published by the map service

Feature Layer	Attribute fields		
	Feature code	Feature Type	Data type
Hydrants	HD	Hydrant	Point
Lamp posts	LP	Lamp post	Point
Manholes	MH	Manhole	Point
Storm Drains	SD	Strom drain	Point
Signs	SI	Sign	Point
Telephone Poles	TP	Telephone poles	Point
Trees	TR	Tree	Point
Walkways	WW	Walkway	Line
Sidewalks	SW	Sidewalk	Line
Crosswalks	CW	Crosswalk	Line
Streets	ST	Street	Line
Fences	FN	Fence	Line
Retaining Walls	RW	Retaining wall	Line
Contours	Contours	Contour lines	Line
Building	BL	Building	Polygon
Parking Lots	PL	Parking lot	Polygon
Bick Racks	BR	Bike rack	Polygon
Benches	BN	Bench	Polygon
Green Fields	GF	Green field	Polygon
Wooded Areas	WA	Wooded area	Polygon
Playgrounds	PG	Playground	Polygon

5.4 Examining the Map Service Performance

The map service evaluated in terms of the data visualization and retrieve. The web service functionalities were examined to illustrate the capabilities of representing the framework specifications. Various examples mentioned below to show the map service performance:

- **The checklist** examined in order to evaluate the visibility controller function. Using the checklist, user can compare and interpret the various types of features existing on the campus and then understand what features are expected to be surveyed for the survey camp. Figure 5.8 shows only the base map with no layers selected in the checklist; on the other hand, figure 5.9 shows the features selected in the checklist along with describing the symbols utilized to represent the features.

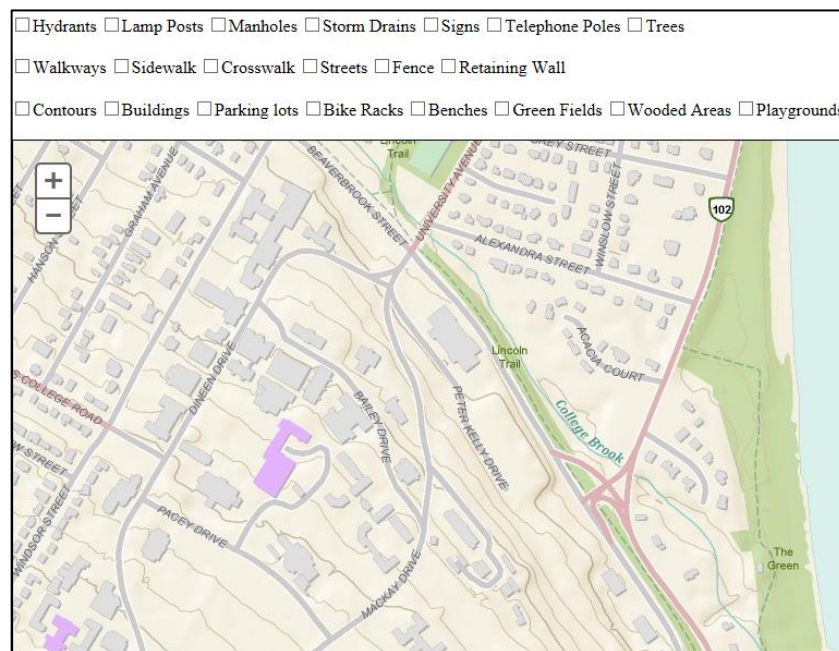


Figure 5.8: Example of the map service with no selected features

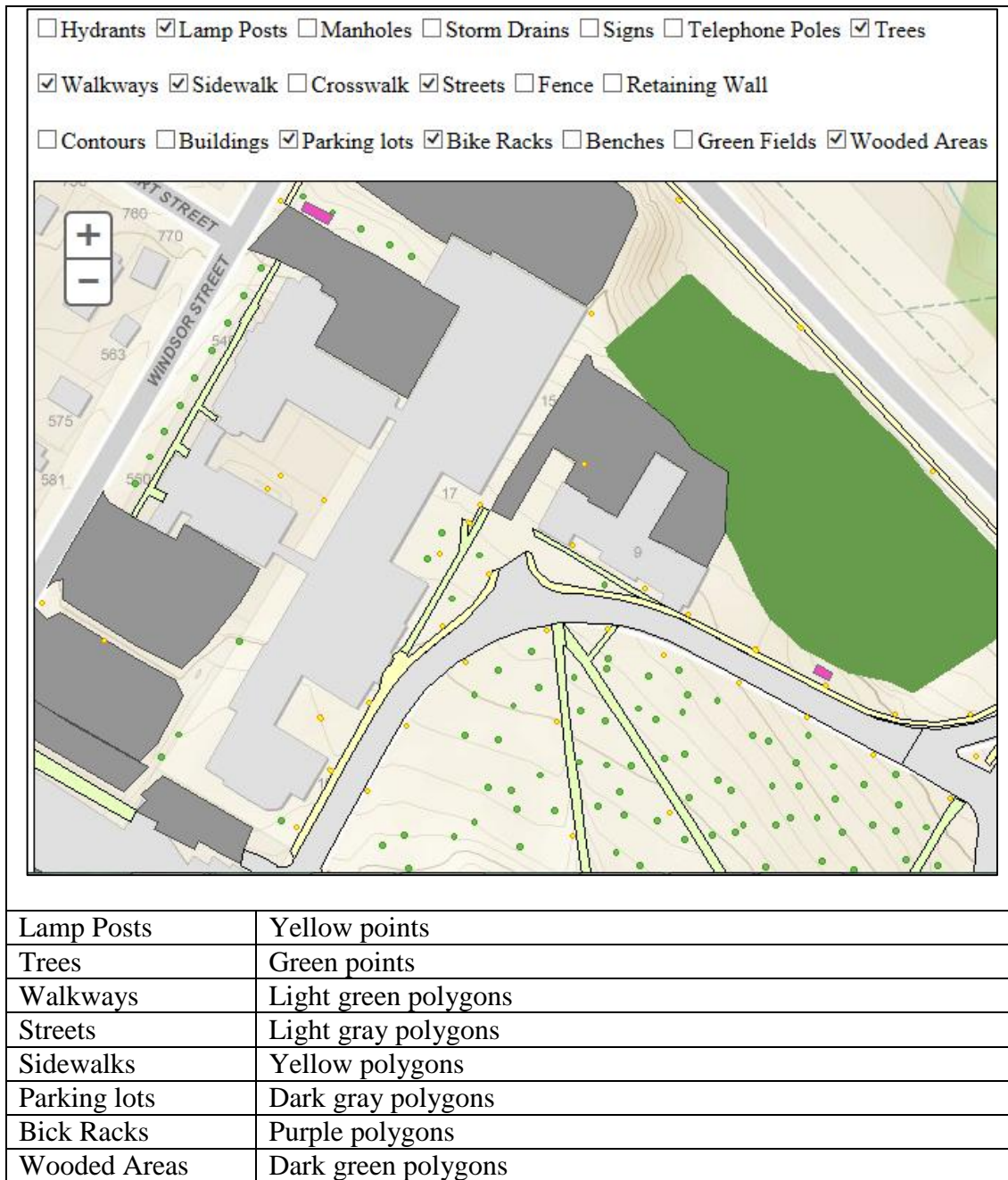


Figure 5.9: Example of representing features selected in the checklist

- *The query capability* was examined in figure 5.10

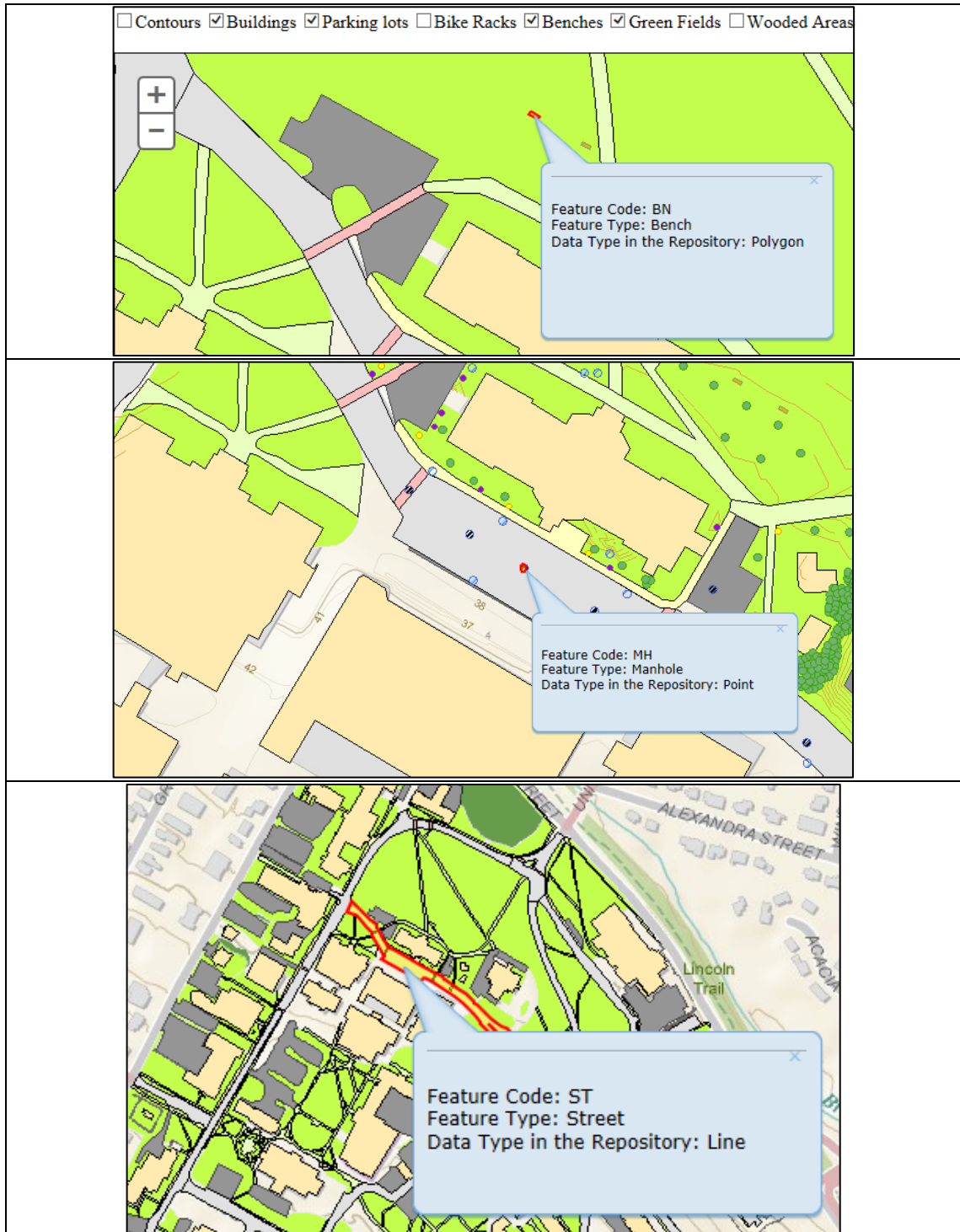


Figure 5.10: Example of querying the features on the map service

5.5 Summary

The map service developed using various ArcGIS JavaScript API classes. The visibility controller and query task functions were added as the major capabilities to the service in order to represent the spatial framework specifications. The JavaScript classes utilized in scripting codes selected according to their ability of data visualizing and retrieving. Furthermore, the factor of integration between the codes was considered to avoid any incompatibility between the JavaScript classes utilized in the code. However, there was an issue with implementing the query task. In order to solve the issue, all the features were converted to polygons and then the correct data type was specified as an attribute for the features.

By using the map service, users understand the repository structure in terms of interpreting the storing features as well as learning the feature coding system expected to implement in the survey camp. Furthermore, the areas on the campus where either surveying is not completed or might need updating with new survey data could be identified.

Chapter 6 Conclusions

The overall goal of this thesis was to design and create a campus GIS via developing a spatial framework for managing the GGE survey camp geospatial holdings. It represented a set of system tools including the geodatabase model and the Web-GIS service by which the framework could manage and organize the available map data. In addition it could define the specifications for the future survey data collections.

The general methodology was illustrated in Chapter 2 along with the characteristics of the systems, technologies, and procedures applied first to analyze the existing map data and then to build the information system tools. The existing map datasets were assessed and analysed in chapter 3 to specify their issues and then to determine the solutions and specifications by which issues could be minimized. Therefore, the existing and future map data could be faster and easier categorized and stored in the repository. After identifying the issues and proposing the appropriate solutions, Chapter 3 illustrated the processes of designing and populating the geodatabase model. Chapter 4 introduced the geodatabase model which was designed in a way which allowed controlling the spatial and non-spatial properties of the map data; therefore, it could satisfy the framework requirements for managing the GGE survey camp geospatial data. Finally, Chapter 5 described the Web-GIS service developed to publish an interactive base map of the UNB campus representing the framework characteristics in terms of the campus key map layers and their attribute codes.

6.1 Research Outcomes and Issues Encountered

The original objectives, outcomes, and conclusions obtained from the thesis are provided bellow:

Objectives and Outcomes:

1. To design and build a geographical repository in order to categorize, store, and manage the GGE survey camp geospatial data.

- Based on what mentioned in the development workflow, after reviewing other research work regarding developing geodatabase models it was found that geodatabases are very appropriate and useful form of geographical repositories. Therefore, designing and building the geodatabase model was the first important and critical part of this project. The geodatabase was created and then used to organize and store the existing survey camp map data. Moreover, a base-map was generated to visually show the content of the data stored in the repository. On the other hand, the Web-GIS was implemented with respect to the data content and their specifications in the repository (e.g. map layers and the associating feature codes).
- Populating the geodatabase was the most difficult part of the building procedure. There were some feature types (e.g. green fields, parking lots, walkways, etc.) that were not included in the source of the map data (CAD files). Therefore, it was necessary to apply manual editing to generate the missing features regarding to their surrounding objects.

- The geodatabase model was mostly created based on the available geospatial information of the GGE survey camp; as a result, any new type of geospatial data in the future map data may not match with the current model. Therefore, the developed geodatabase model may partially satisfy the goal of managing and storing the future geospatial data.

2. To define the spatial and non-spatial characteristics of the spatial reference framework in order to establish the validation methods as well as defining new specifications for the survey camp operation in terms of the feature surveying and coding(which all result in facilitating the repository maintenance process).

- It was tried to define a series of characteristics for the framework to minimize inconsistency issues. The consistency in feature surveying, spatial representation forms, and the feature coding schema were found as the most effective factors for satisfying this goal. Additionally, these factors were found essential and helpful for validating the data. Performance of the maintenance tools is depended on validity of the input data; as a result, validation processes must be applied before applying the maintenance tools. That, therefore, any invalid input data might cause difficulties for the maintenance processes and tools (GIS models) in terms of performance and results.

3. *To develop a Web-GIS service to publish an interactive base map of the UNB campus in order to represent the spatial framework characteristics and specifications.*

- This objective was implemented as the last outcome of this project. It completely satisfied the goal of publishing an interactive base map of the UNB campus on the web. The interactive base map allows visually comparing, interpreting, and querying the geospatial content of the survey camp. These capabilities of the service could facilitate the process of familiarizing the users with the framework specifications. Furthermore, this map service can be introduced as one of the major outcomes of a campus GIS to help the users interpret the campus area.
- The difficulties in developing the Web-GIS have been successfully resolved by providing appropriate functions for visualizing and querying the map layers and features as well as representing the data with the appropriate spatial forms on the map service (i.e. providing the map layers with polygonal forms).

6.2 Recommendations for Future Research

1. Building a geodatabase model for a specific type of geospatial data must be implemented with respect to the data content that are being stored in the repository. The type of data and why they must be stored in a geodatabase must be considered before selecting any approach for implementing the geodatabase. The geodatabase model designed and implemented must be capable of controlling and managing the key spatial and non-spatial behaviors and characteristics of the data.
2. It is very important to define and specify the most efficient methods for maintaining the repository. By introducing the maintenance methods, the repository will be kept updated and useful. Additionally, the maintaining methods must have the capability of categorizing, and analysing the new type of data.
3. To develop a Web-GIS service, the most important factor is to choose the most appropriate technologies and systems in terms of the server and client sides. With respect to the purpose of using a Web-GIS service, the service functionalities must be determined and then the suitable technologies could be utilized to obtain the best results.

4. To develop a geospatial reference framework via integration of a set of information products (e.g. geographical repository models and mapping services), it is very important to consider the consistency factor in the framework characteristics. For example, data types, representation forms, storage structure, and identification methods are critically influence the development procedure. If a set of consistence characteristics could be successfully defined for the systems; respectively, a framework will be developed with the capabilities of providing effective and sufficient management and retrieval methods for the geospatial data.
5. Developing a method that allows assessing data accuracy will improve the consistency of the information model. That, therefore, will improve the reliability of the model in terms of data validity, and respectively facilitate the data management processes as well.

Bibliography

- ArcGIS API for JavaScript. Retrieved from
<https://developers.arcgis.com/en/javascript/jsapi/>
- ArcGIS resources. Retrieved from
<http://resources.arcgis.com/en/help/main/10.1/index.html#/0154000002np000000>
- ArcGIS server REST API. Retrieved from
<http://resources.arcgis.com/en/help/rest/apiref/>
- Arctur D. and Zeiler M. (2004). *"Designing geodatabase: Case studies in GIS data modeling"*. United States of America: ESRI, Inc.
- Binh T. Q., Huan N. C., Thuy L. P. (2008). *"DESIGNING A GEODATABASE MODEL FOR URBAN INFORMATION SYSTEM AT THE BASIC LEVEL (case study ill nguyell du ward, hai ba trllng district, hanoi city)"*. (Annual Report of FY 2007, The Core University Program between Japan Society for the Promotion of Science (JSPS) and Vietnamese Academy of Science and Technology (VAST)). Osaka University.
- Charlyne T. Smith and Hugh A. Devine. (2006). "The personal geodatabase as a bmp for stormwater management". *Proceedings of the 5th Southern Forestry and Natural Resources GIS Conference*, Warnell School of Forestry and Natural Resources, University of Georgia, Athens, GA. pp.82-90.
- Chen B., G. Y. (2010). "The building of network geographic information system based on arc GIS". *International Conference on Computer Application and System Modeling*, , 14 PP.90-93. doi: 10.1109/ICCASM.2010.5622400
- Chesnaux R, Lambert M, Walter J, Fillastre U, Hay M, Rouleau A., (2011). "Building a geodatabase for mapping hydrogeological features and 3D modeling of ground water systems: Application to the Saguenay–Lac-st.-jean region, Canada". *Computers & Geosciences*, 37(11), 1870–1882. doi: 10.1016/j.cageo.2011.04.013
- Deliiska B., D. M., Milchev R. (2008). "Campus GIS functionality and interface". *International Scientific Conference Computer Science*, , 1 PP.291-295.
- Fangli N., Kang W., Juan W. (2010). "Designing and realization of campus WebGIS based on ArcGIS server". *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE)*, Changchun. , 5 PP.72-75.

- Gosain A. K., Rao S., Singh P., Arora A. (2010). "Integration of bio-geo spatial database for selected watersheds in himalayan region". *Current Science*, VOL. 98, NO. 2, PP.183-191.
- HUANG J., Zhan Y., CUI W., YUAN Y., QI P. (2010). "Development of a campus information navigation system based on GIS". *2010 International Conference on Computer Design and Applications*, Qinhuangdao. , 5 PP.491-494.
- Kalman N. B., Hogle I. B., Viers J. H. (2004,). "Designing a geodatabase for your project: A wetlands delineation example". Message posted to http://escholarship.ucop.edu/uc/jmie_ice_icepubs
- Lifeng Y. (2010). "Design and accomplishment of campus WebGIS based on ArcIMS A case study on nanjing university of posts and telecommunications". *International Conference on Computer and Communication Technologies in Agriculture Engineering*, Chengdu. , 1 PP.44-46.
- Local government information model (ArcGIS 10). (2013). Retrieved from <http://www.arcgis.com/home/item.html?id=b8905e21104342afbe830da28d11b2b9>
- Lua H., Nihonga W., Changb W., Yujiaa C. (2010). "The research on the WebGIS application based on the J2EE framework and ArcGIS server". *International Conference on Intelligent Computation Technology and Automation*, Changsha. , 3 PP.942-945.
- Mathiyalagana V., Grunwaldb S., Reddyb K. R., Bloomb S. A. (2005). "A WebGIS and geodatabase for Florida's wetlands". *Computers and Electronics in Agriculture*, 47, PP.69-75. doi: 10.1016/j.compag.2004.08.003
- Mortada K. A., Mohamed K., Abdellah M., Abderrahmane M. (2011). "a descriptive model for developing a hydraulic geodatabase by using the GIS software". *International Journal of Computer Science & Information Technology (IJCSIT)*, Vol 3, No 2, pp.177-189. doi: 10.5121/ijcsit.2011.3213
- Silayo E. H. (2012). "Cartography in a gis environment". *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, , XXXIV PP.106-112.
- Tsouchlaraki A., Achilleos G., Nikolidakis A., Nasioula Z. (2010). Building a geodatabase for urban road network environmental quality. *3rd International Conference on Cartography and Gis*, Nessebar, Bulgaria.

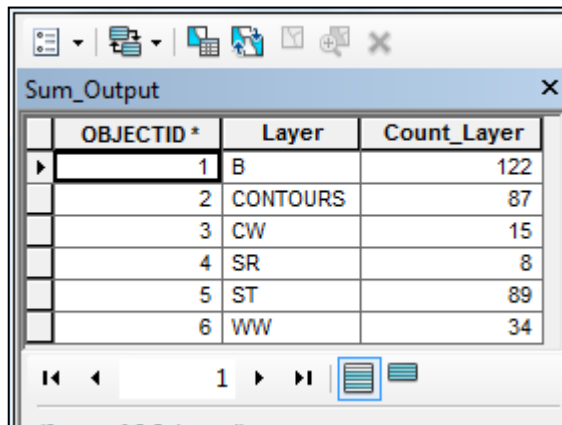
- Wang J.,Stein T. C.,Heet T.,Scholes D. M.,Arvidson R. E. (2010). "A WebGIS for apollo analyst's notebook". *Second International Conference on Advanced Geographic Information Systems, Applications, and Services*, St. Maarten. PP.88-92. doi: 10.1109/GEOProcessing.2010.20
- Yang Y.,Xu J.,Zheng J.,Lin S. (2009). "Design and implementation of campus spatial information service based on google maps". *International Conference on Mnagement and Service Science (MASS)*, Wuhan. PP.1-4.
- Zeiler M. (1999). *"Modeling our world the ESRI guide to geodatabase design"*. United States of America: Environmental Systems Research Institute, Inc.
- Zhao J.,Wang Q.,Wang Y.,Yang W. (2010). "Real estate early warning information release platform based on ArcGIS server". *International Conference on E-Business and E-Government*, Guangzhou. PP.2308 - 2311.
- Zhong S., B. T.,Wang Z. (2010). "Management information system for provincial mineral resources based on ArcGIS". *2nd Conference on Environmental Science and Information Application Technology*, Wuhan. , 1 PP.325 - 328. doi: 10.1109/ESIAT.2010.5568570

Appendix I Geodatabase Maintenance

I.1 Validating the Feature Codes

A sample data of the survey camp was used in this section in order to demonstrate the procedure of validating the new data. The sample data was first analyzed with validating methods and then the issues were outlined and eliminated. After eliminating the issues, the sample data was again examined and then the valid features were identified. The steps mentioned below describe the validating process.

1. *Summarizing the feature attribute codes* to start the validating process. The *Layer* field was summarized to quickly examine the feature coding structure that utilized in the survey camp and mapping procedures. Summarizing the attribute table provided an overview of the codes that utilized in the *Layer* field. Figure I.1 shows the attribute table of the sample data summarized in ArcMap.

The image shows a screenshot of the ArcMap interface, specifically the 'Summarize' tool's output window titled 'Sum_Output'. The window displays a summary table with three columns: 'OBJECTID *', 'Layer', and 'Count_Layer'. The table contains six rows of data. The first row is highlighted with a blue selection bar. The table data is as follows:

OBJECTID *	Layer	Count_Layer
1	B	122
2	CONTOURS	87
3	CW	15
4	SR	8
5	ST	89
6	WW	34

Figure I.1: Summarization of the attributes codes in the Layer field

2. *Examining the feature codes* utilized in the *Layer* field. As illustrated in figure I.1, there are six different codes included in the *Layer* field. The sample data attribute table were examined against the domain table to specify the matches according to the domain coding system proposed in the geodatabase design. Figure I.2 shows some of the valid codes in the domain table.

OID *	Codes	Description
1	BL	Buildings
2	BS	Building Stairs
3	BN	Benches
4	Contours	Contours
5	CB	Curbs
6	CW	Crosswalks
7	EB	Electrical Boxes
8	FN	Fences
9	GF	Green Fields
10	GD	Guardrails
11	HD	Hydrants
12	LI	Lights
13	LP	Lamp Posts
14	MH	Manholes
15	PL	Parking lots
16	SI	Signs
17	SR	stairs
18	ST	street
19	SD	Storm drains
20	SW	Sidewalk
21	WA	Wooded areas
22	WW	Walkways
23	TR	Tree

Figure I.2: Domain table

The Layer field in the sample data was joined with the Codes field in the domain table. The features in the sample data which their attributes in the *Layer* field matched with the attributes in Codes field were specified after the join. After the join, the matching features obtained the attributes of the Code and Description fields in the domain table. As a result, the features which did not match obtained “Null” values and then were specified as wrong attribute codes. To specify the invalid codes, the *Layer* field in the sample data was summarized with respect to the *Codes* and *Description* fields. The summarizing table illustrated in the figure I.3 indicating the empty rows in the *First_Codes* and *First_Description* fields along with the corresponding invalid attribute codes in the *Layer* field. As illustrated, the B, CONTOUR, and TRAVERS attributes were determined as invalid codes in this case.

OBJECTID *	Layer	Count_Layer	First_Codes	First_Description
1	B	122		
2	CONTOUR	87		
3	CW	15	CW	Crosswalks
4	SR	8	SR	stairs
5	ST	89	ST	street
6	TRAVERS	9		
7	WW	34	WW	Walkways

Figure I.3: Example of the invalid codes

Since the Codes in the *Layer* field are utilized in the maintaining process, the invalid codes must be revised and corrected prior to starting the process. In fact, the invalid codes must be replaced with the valid codes. The invalid codes were selected by SQL query and then replaced with the valid codes according to the type of features selected in the query. In this case, the B and CONTOUR codes were respectively replaced with BL for the buildings and Contours for the contour lines.

I.2 Validating the Features' Data Type

For validating the features' data type, the sample data were analyzed, edited and then imported as feature classes into the datasets. The process of validating the features' data type is demonstrated in the steps mentioned below:

1. *Examining the features' type and class* with respect to the attributes' descriptions associated with the objects. For example, the objects classified as buildings in the sample data include other type of features which must be classified as lampposts. Figure I.4 shows the lamppost (light) features coded as buildings.

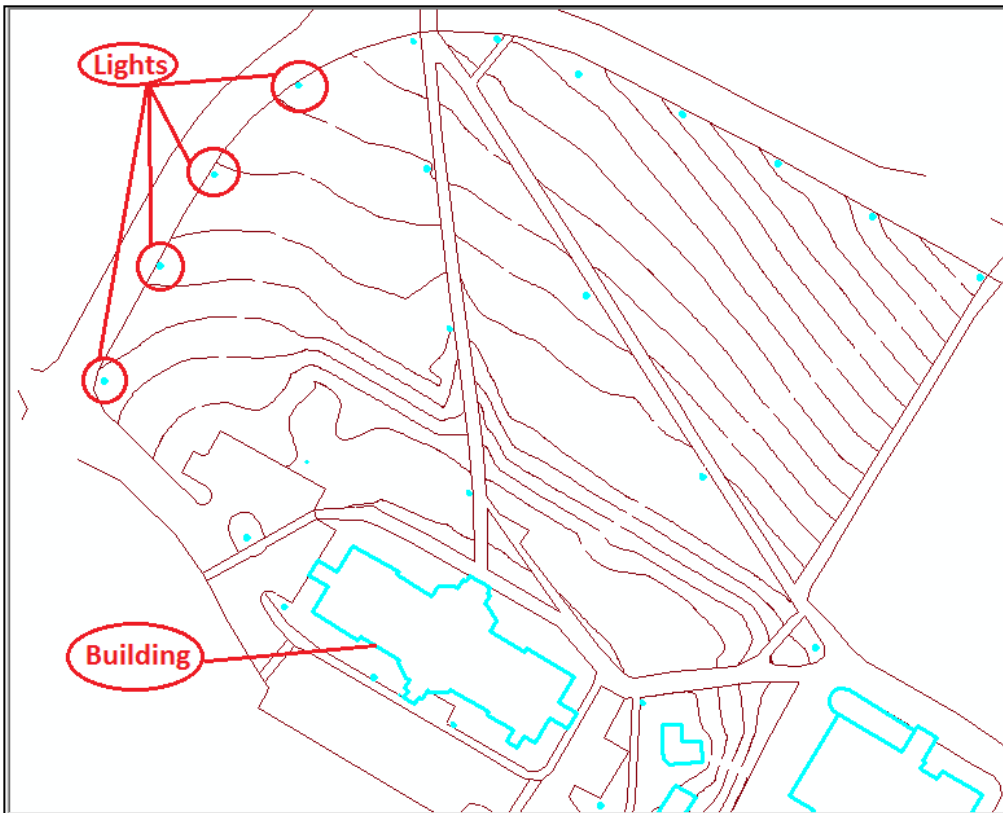


Figure I.4: Example of the wrong feature coding

Manual editing applied to reclassify the objects with the correct classes. The attribute table was again examined against the domain table to make sure the codes are matched with the descriptions. Figure I.5 shows the attribute table revised and corrected. The same validation process must perform for the point layers as well.

	OBJECTID *	Layer	Count_Layer	OID *	Codes *	Description
▶	1	BL	96	1	BL	Buildings
	2	Contours	87	4	Contours	Contours
	3	CW	15	6	CW	Crosswalks
	4	LP	26	13	LP	Lamp Posts
	5	SR	8	17	SR	stairs
	6	ST	89	18	ST	street
	7	WW	34	22	WW	Walkways

Figure I.5: Example of the validated attributes

2. *Examining the features' spatial data type* against the corresponding feature classes specified in the design phases and created in the datasets. For example, the lamppost features stored as a point feature class in the geodatabase; thus, the lampposts in the sample data must be point features as well. However, because the original point data source was not available for this sample data then the lamppost points created with geoprocessing tools. In this case, the *Feature to Point* tools utilized to generate the lamppost points; in fact, a point feature class generated from the representative points of the lamppost polylines.

I.3 Applying GIS models

1. *Adding the sample data into the datasets.* Since all the codes validated in the previous steps, a GIS model could utilize first to specify the features by the SQL queries and then to add them as feature classes to the datasets. Moreover, the

editing tool in ArcMap utilized to create some of the feature types such as the green fields and parking lots that were not specified in the original sample data. On the other hand, if an appropriate feature coding was applied to those features with respect to coding system mentioned in the chapter 3, it was possible to easily import the green field and parking lot feature classes into the datasets. Figure I.6 shows the GIS model creates the feature classes.

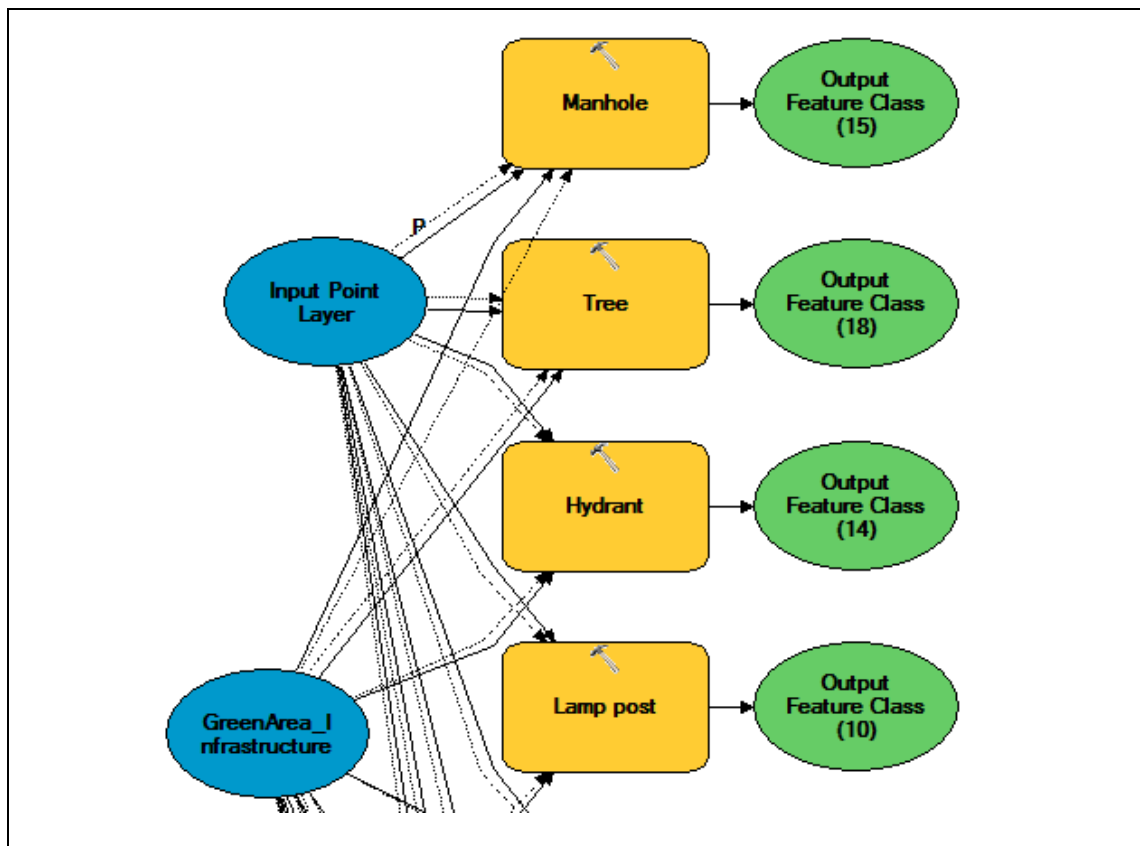


Figure I.6: Example of the GIS model creating feature classes

In the figure I.6, the tools for creating the manhole, tree, hydrant, and lamppost feature classes represented along with the aspects for specifying the input point layer data and the dataset which the output feature classes will be stored in.

2. *Updating the old features* with the corresponding new features or in fact with the new feature classes that added to the datasets in the previous step. Another GIS model used here to spatially join the old and new features to determine the old features that must be replaced with the new one. In this example, the old building layer will be updated with the new building polygons. Figure I.7 shows the old and new layers.



Figure I.7: Example of old and new building polygons

The GIS model that performs the spatial join and updating process represented in the the figure I.8 .

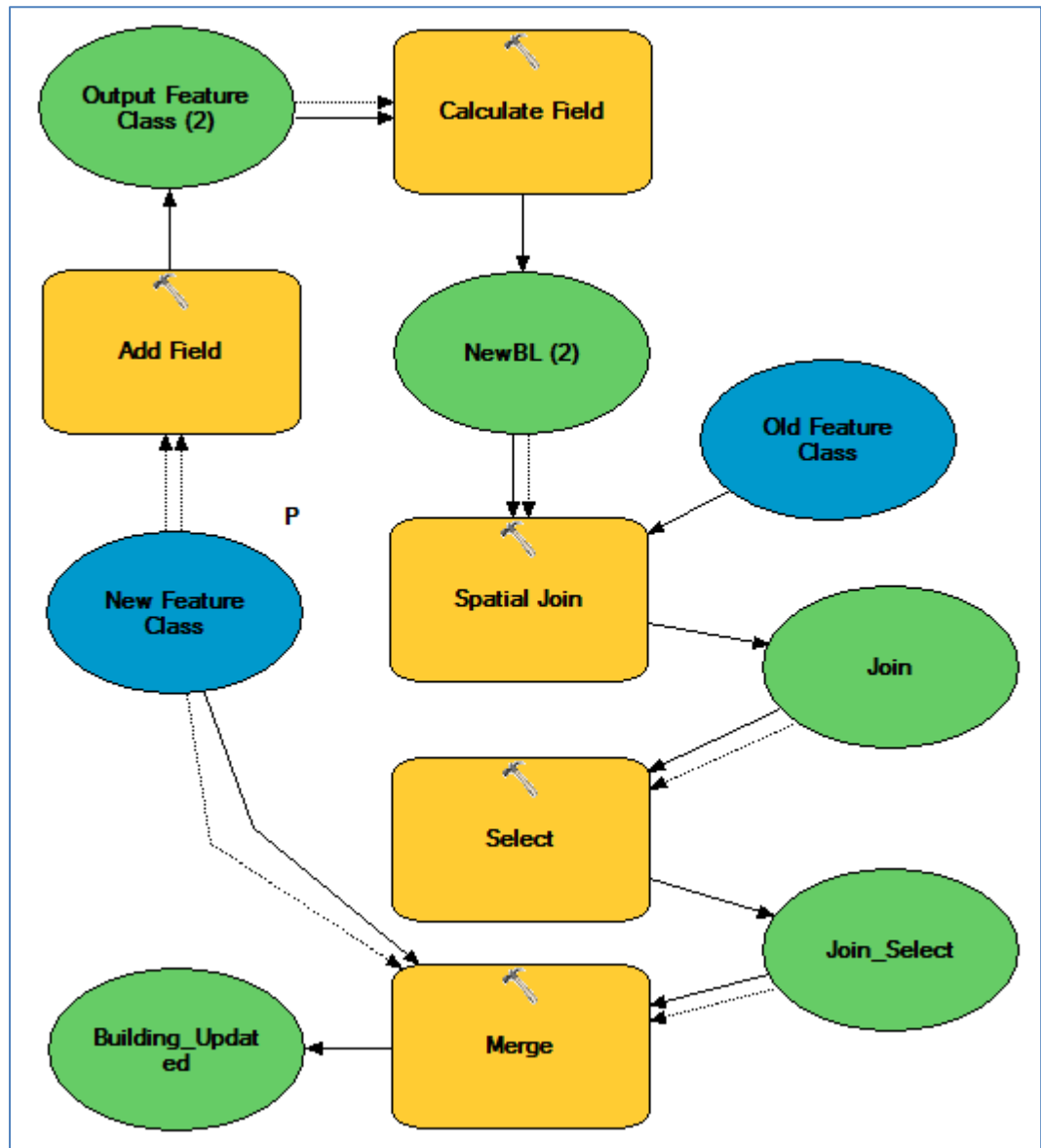


Figure I.8: Example of the GIS model performs the updating process

The results illustrated in the figure I.9 bellow were obtained from the geoprocessing tools of the GIS model shown above.

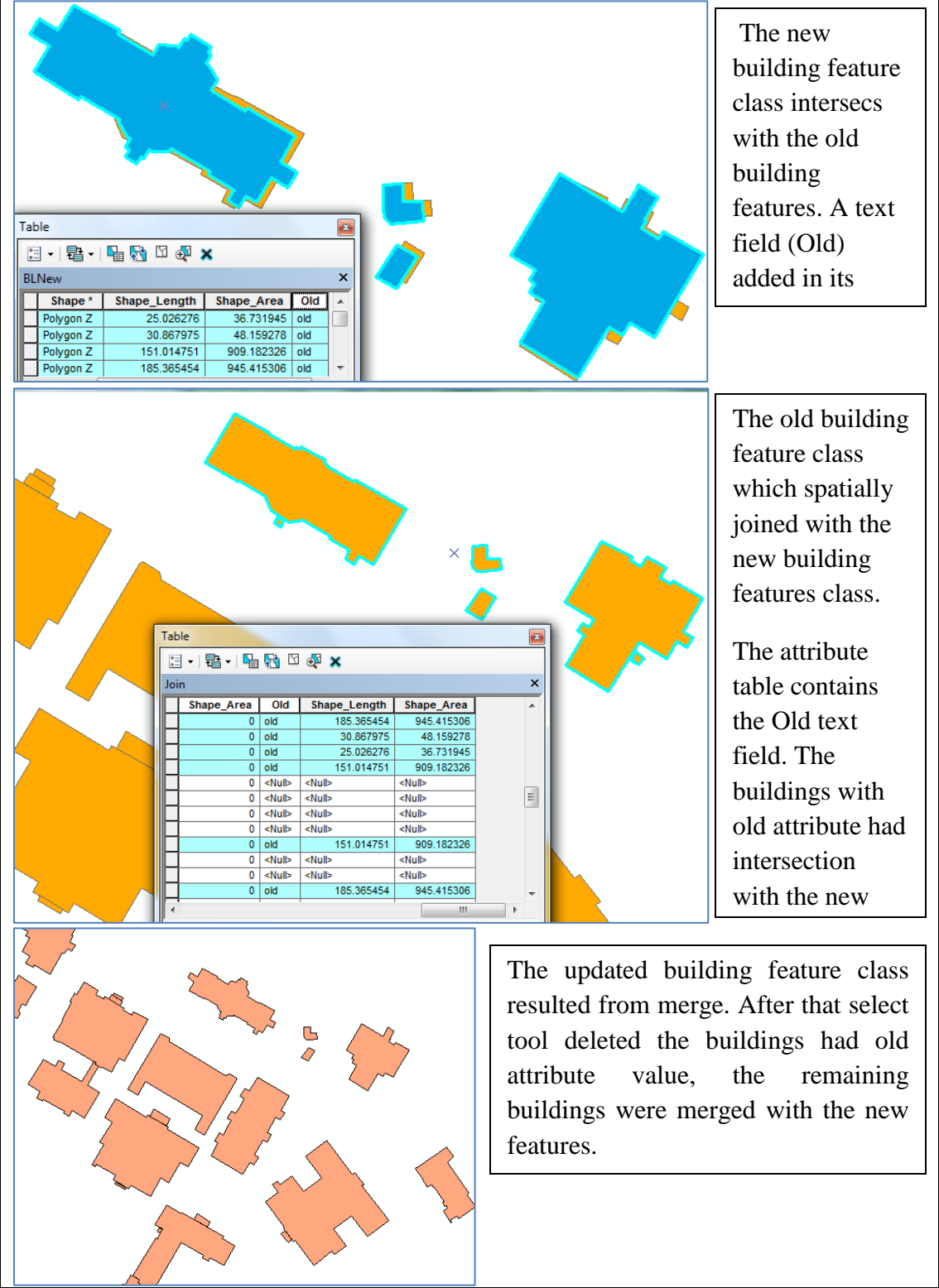


Figure I.9: Examples of the results for the geoprocessing steps of the GIS model

I.4 Glossary of GIS Terms

Attribute table: A tabular file containing information about a set of geographic features, usually arranged so that each row represents a feature and each column represents one feature attribute.

Attribute domain: In a geodatabase, a mechanism for enforcing data integrity. Attribute domains define what values are allowed in a field in a feature class or non-spatial attribute table.

Attribute data: Tabular data that describe the geographic characteristics of features.

Attribute query: A request for records of feature in a table based on their attribute value.

CAD file: The digital equivalent of a drawing, figure, or schematic created using a CAD system. CAD files are the data source for CAD drawing datasets, feature datasets and feature classes.

CAD layer: A layer that references a set of CAD data. CAD data is vector data of a mixed feature type. CAD layers may be of two types: CAD drawing dataset layers, in which one map layer represents the entire CAD file, and CAD feature layers, in which data is organized by geometry type.

Coded value domain: A type of attribute domain that defines a set of permissible values for an attribute in a geodatabase. A coded value domain consists of a code and its equivalent value.

Feature: A representation of a real-world object on a map.

Feature Class: In ArcGIS, a collection of geographic features with the same geometry type (such as point, line, or polygon), the same attributes, and the same spatial reference.

Feature Dataset: In ArcGIS, a collection of feature classes stored together that share the same spatial reference; that is, they share a coordinate system, and their features fall within a common geographic area. Feature classes with different geometry types may be stored in a feature dataset.

Field: A column in a table that stores the values for a single attribute.

Geodatabase: A database or file structure used primarily to store, query, and manipulate spatial data. Geodatabases store geometry, a spatial reference system, attributes, and behavioral rules for data. Various types of geographic datasets can be collected within a geodatabase, including feature classes, attribute tables, raster datasets, network datasets, topologies, and many others.

Geodatabase data model: The schema for the various geographic datasets and tables in an instance of a geodatabase. The schema defines the GIS objects, rules, and relationships used to add GIS behavior and integrity to the datasets in a collection.

Layer: The visual representation of a geographic dataset in any digital map environment.

Line feature: A map feature that has length but not area at a given scale, such as a river on a world map or a street on a city map.

Point feature: A map feature that has neither length nor area at a given scale, such as a city on a world map or a building on a city map.

Polygon feature: In ArcGIS software, a digital map feature that represents a place or thing that has area at a given scale. A polygon feature may have one or more parts. For example, a building footprint is typically a polygon feature with one part.

Polyline feature: In ArcGIS software, a digital map feature that represents a place or thing that has length but not area at a given scale. A polyline feature may have one or more parts. For example, a stream is typically a polyline feature with one part.

Query expression: A type of expression that evaluates to a Boolean (true or false) value, that is typically used to select those rows in a table in which the expression evaluates to true. Query expressions are generally part of a SQL statement.

Tabular data: Descriptive information, usually alphanumeric, that is stored in rows and columns in a database and can be linked to spatial data.

Topology: In geodatabases, the arrangement that constrains how point, line, and polygon features share geometry.

Topology rule: An instruction to the geodatabase defining the permissible relationships of features within a given feature class or between features in two different feature classes.

Appendix II Geodatabase Schema Documentation

In documenting a geodatabase, first an overview of the structure must be provided.

Figure II.1 shows the geodatabase model structure that was developed in this thesis.

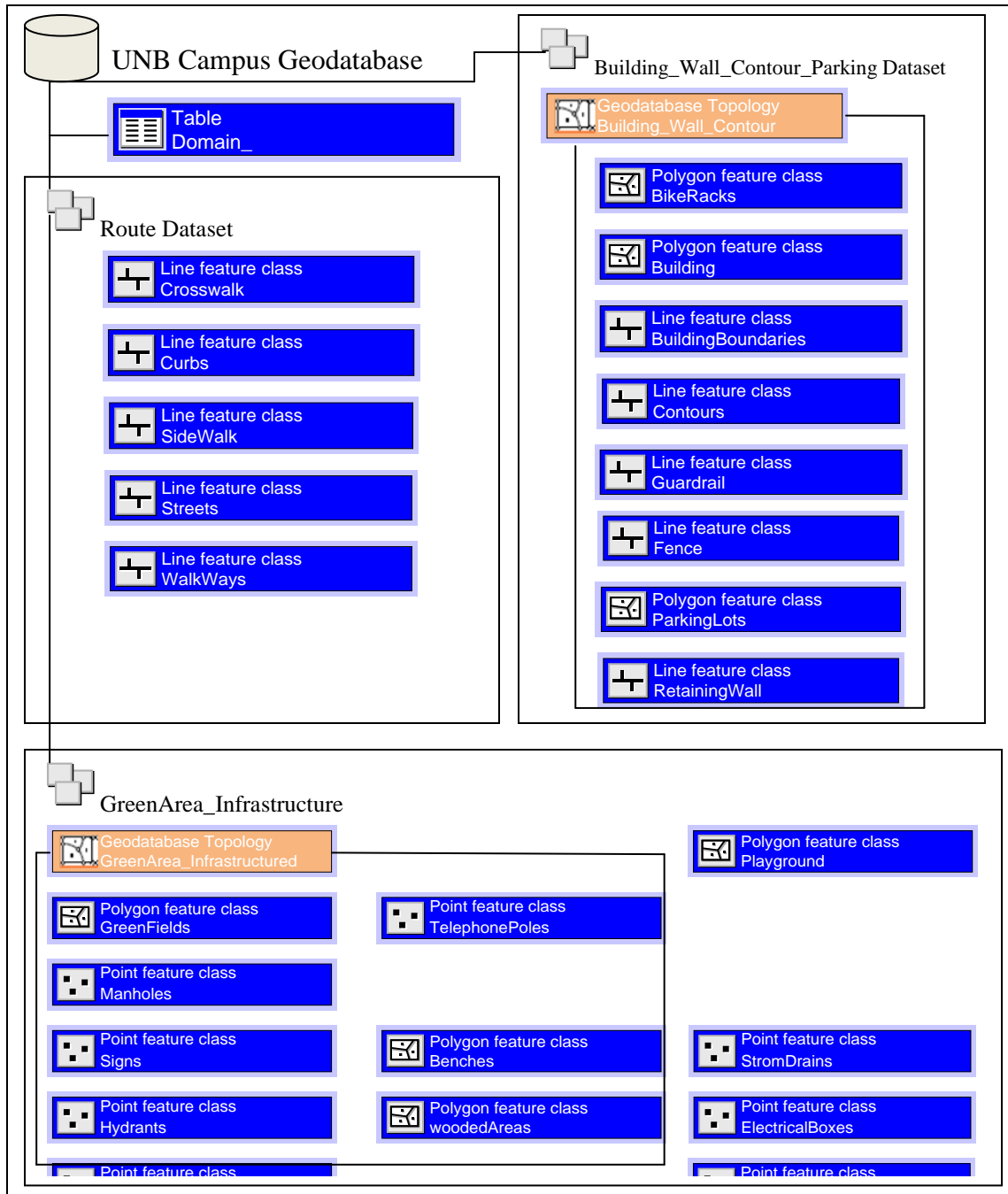


Figure II.1: Overview of UNB campus geodatabase structure

After showing the overview of the geodatabase structure, each dataset must be illustrated separately. In this illustration, each feature class is documented in order to show the tabular content importantly including: field name, data type, and domain. Figures II.2, II.3, II.4 show the feature classes in datasets along with the tabular contents.

Route Dataset										
Simple feature class Streets							Geometry	Polyline	Contains M values	No
							Contains Z values	No		
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length			
OBJECTID	Object ID									
SHAPE	Geometry	Yes								
SHAPE_Length	Double	Yes			0	0				
FeatureCode	String	Yes						2		
FeatureType	String	Yes		Codes				50		
Simple feature class WalkWays							Geometry	Polyline	Contains M values	No
							Contains Z values	No		
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length			
OBJECTID	Object ID									
SHAPE	Geometry	Yes								
SHAPE_Length	Double	Yes			0	0				
FeatureCode	String	Yes						2		
FeatureType	String	Yes		Codes				50		
Simple feature class Crosswalk							Geometry	Polyline	Contains M values	No
							Contains Z values	No		
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length			
OBJECTID	Object ID									
SHAPE	Geometry	Yes								
SHAPE_Length	Double	Yes			0	0				
FeatureCode	String	Yes						2		
FeatureType	String	Yes		Codes				50		
Simple feature class SideWalk							Geometry	Polyline	Contains M values	No
							Contains Z values	No		
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length			
OBJECTID	Object ID									
SHAPE	Geometry	Yes								
SHAPE_Length	Double	Yes			0	0				
FeatureCode	String	Yes						2		
FeatureType	String	Yes		Codes				50		
Simple feature class Curbs							Geometry	Polyline	Contains M values	No
							Contains Z values	No		
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length			
OBJECTID	Object ID									
SHAPE	Geometry	Yes								
SHAPE_Length	Double	Yes			0	0				
FeatureCode	String	Yes						2		
FeatureType	String	Yes		Codes				50		

Figure II.2: Route dataset

Building_Wall_Contour_Parking Dataset									
	Simple feature class BikeRacks							Geometry Contains M values Contains Z values	Polygon No No
Field name	Data type	Allow nulls	Default value	Domain	Prec- ision	Scale	Length		
OBJECTID	Object ID								
SHAPE	Geometry	Yes							
SHAPE_Length	Double	Yes			0	0			
SHAPE_Area	Double	Yes			0	0			
FeatureCode	String	Yes							2
FeatureType	String	Yes		Codes					50
	Simple feature class BuildingBoundaries							Geometry Contains M values Contains Z values	Polyline No No
Field name	Data type	Allow nulls	Default value	Domain	Prec- ision	Scale	Length		
OBJECTID	Object ID								
SHAPE	Geometry	Yes							
FID_Building	Long integer	Yes			0				
FeatureCode	String	Yes							2
FeatureType	String	Yes		Codes					50
SHAPE_Length	Double	Yes			0	0			
	Simple feature class Building							Geometry Contains M values Contains Z values	Polygon No No
Field name	Data type	Allow nulls	Default value	Domain	Prec- ision	Scale	Length		
OBJECTID	Object ID								
SHAPE	Geometry	Yes							
SHAPE_Length	Double	Yes			0	0			
SHAPE_Area	Double	Yes			0	0			
FeatureCode	String	Yes							2
FeatureType	String	Yes		Codes					50
	Simple feature class Contours							Geometry Contains M values Contains Z values	Polyline No No
Field name	Data type	Allow nulls	Default value	Domain	Prec- ision	Scale	Length		
OBJECTID	Object ID								
SHAPE	Geometry	Yes							
SHAPE_Length	Double	Yes			0	0			
FeatureCode	String	Yes							10
FeatureType	String	Yes							50
	Simple feature class Guardrail							Geometry Contains M values Contains Z values	Polyline No No
Field name	Data type	Allow nulls	Default value	Domain	Prec- ision	Scale	Length		
OBJECTID	Object ID								
SHAPE	Geometry	Yes							
SHAPE_Length	Double	Yes			0	0			
FeatureCode	String	Yes							4
FeatureType	String	Yes		Codes					50
	Simple feature class Fence							Geometry Contains M values Contains Z values	Polyline No No
Field name	Data type	Allow nulls	Default value	Domain	Prec- ision	Scale	Length		
OBJECTID	Object ID								
SHAPE	Geometry	Yes							
SHAPE_Length	Double	Yes			0	0			
FeatureCode	String	Yes							2
FeatureType	String	Yes		Codes					50
	Simple feature class RetainingWall							Geometry Contains M values Contains Z values	Polyline No No
Field name	Data type	Allow nulls	Default value	Domain	Prec- ision	Scale	Length		
OBJECTID	Object ID								
SHAPE	Geometry	Yes							
SHAPE_Length	Double	Yes			0	0			
FeatureCode	String	Yes							2
FeatureType	String	Yes		Codes					50

Figure II.3: Building_Wall_Contour_Parking Dataset



GreenArea_Infrastructure Dataset

Simple feature class		Geometry		Polygon			
GreenFields		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
SHAPE_Length	Double	Yes			0	0	
SHAPE_Area	Double	Yes			0	0	
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Point			
Signs		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Point			
Hydrants		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Point			
ElectricalBoxes		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
FeatureCode	String	Yes					4
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Point			
TelephonePoles		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Point			
LampPosts		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Polygon			
Playground		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
SHAPE_Length	Double	Yes			0	0	
SHAPE_Area	Double	Yes			0	0	
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Polygon			
woodedAreas		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
SHAPE_Length	Double	Yes			0	0	
SHAPE_Area	Double	Yes			0	0	
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Polygon			
Benches		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
SHAPE_Length	Double	Yes			0	0	
SHAPE_Area	Double	Yes			0	0	
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Point			
Trees		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Point			
StromDrains		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Simple feature class		Geometry		Point			
Manholes		Contains M values		No			
		Contains Z values		No			
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
FeatureCode	String	Yes					2
FeatureType	String	Yes		Codes			50

Figure II.4: GreenArea_Infrastructure Dataset

After documenting the datasets and feature classes, tables in geodatabase must be illustrated. A domain table was created in the geodatabase which contains two fields, the domain coded value and their descriptions. The coded values in the table are used for data validation. Figure II.5 shows the domain table.

Coded value domain	
Codes	
Description	Description
Field type	String
Split policy	Default value
Merge policy	Default value
Code	Description
BL	Buildings
BS	Building Stairs
BN	Benches
Contours	Contours
CB	Curbs
CW	Crosswalks
EB	Electrical Boxes
FN	Fences
GF	Green Fields
GD	Guardrails
HD	Hydrants
LI	Lights
LP	Lamp Posts
MH	Manholes
PL	Parking lots
SI	Signs
SR	stairs
ST	street
SD	Storm drains
SW	Sidewalk
WA	Wooded areas
WW	Walkways
TR	Tree
VP	VPort
TP	Telephone poles
RW	Retaining walls
GFST	Green fields_ Streets
GFWW	Green fieds_ Walkways
PLST	Parking lots_ Streets
PLSW	Parking lots_ Sidewalks
BLPL	Buildings_ Parking lots
BLGF	Buildings_ Green fields
GFPL	Green fields_ Parking lots
GFSW	Green fields_ Sidewalks
STGF	Streets_ Greenfields
WWGF	Walkways_ Green fields
STPL	Streets_ Parking lots
PLBL	Parking lots_ Buildings
GFBL	Green fields_ Buildings
STBL	Streets_ Buildings
PLGF	Parking lots_ Green fields
SWGf	Sidewalks_ Greenfields

Figure II.5: Domain table

Appendix III Web Service User Guide and Code

III.1 User Guide

Figure II.1 shows the ESRI World Topographic background base map. As it is shown in the figure, the UNB campus area is represented once the map service is loaded in the internet browser. The checkbox on the top indicate the feature layers to be selected for display.

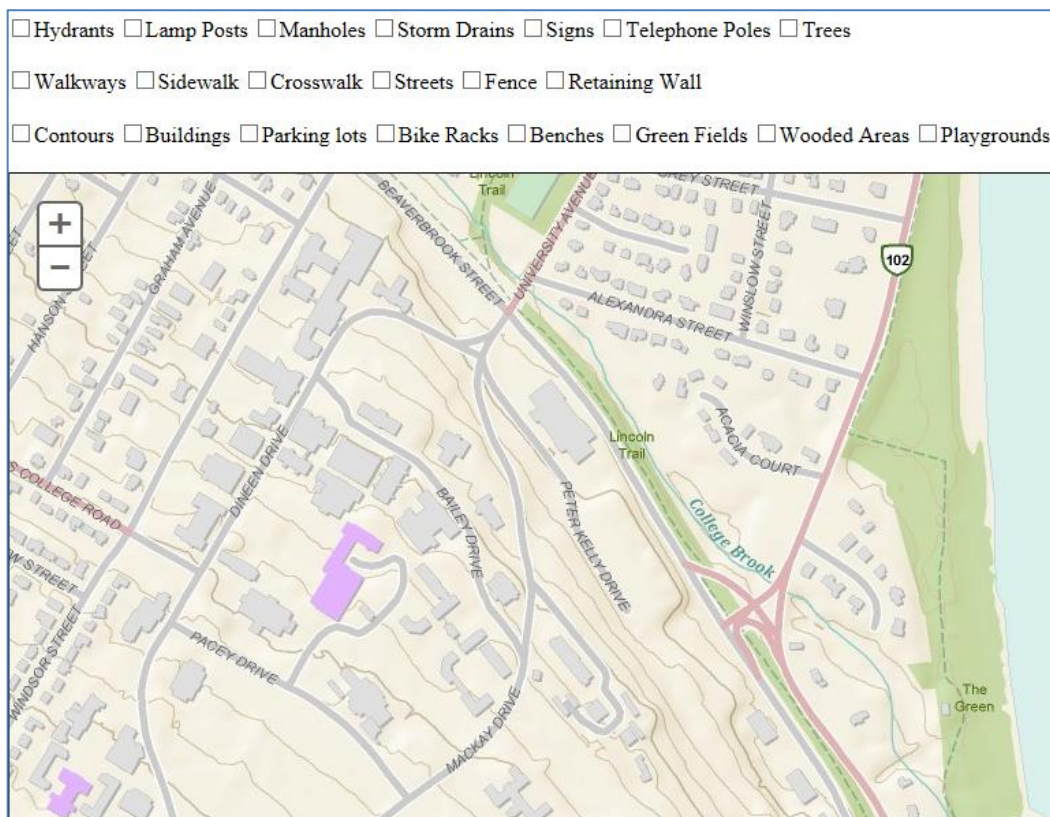


Figure III.1: Example of map service representing the background base map

Figure II.2 shows the map service displaying some layers that have been selected with the checkboxes. This capability helps to interpret and explore the features distribution of each individual layer around the campus.

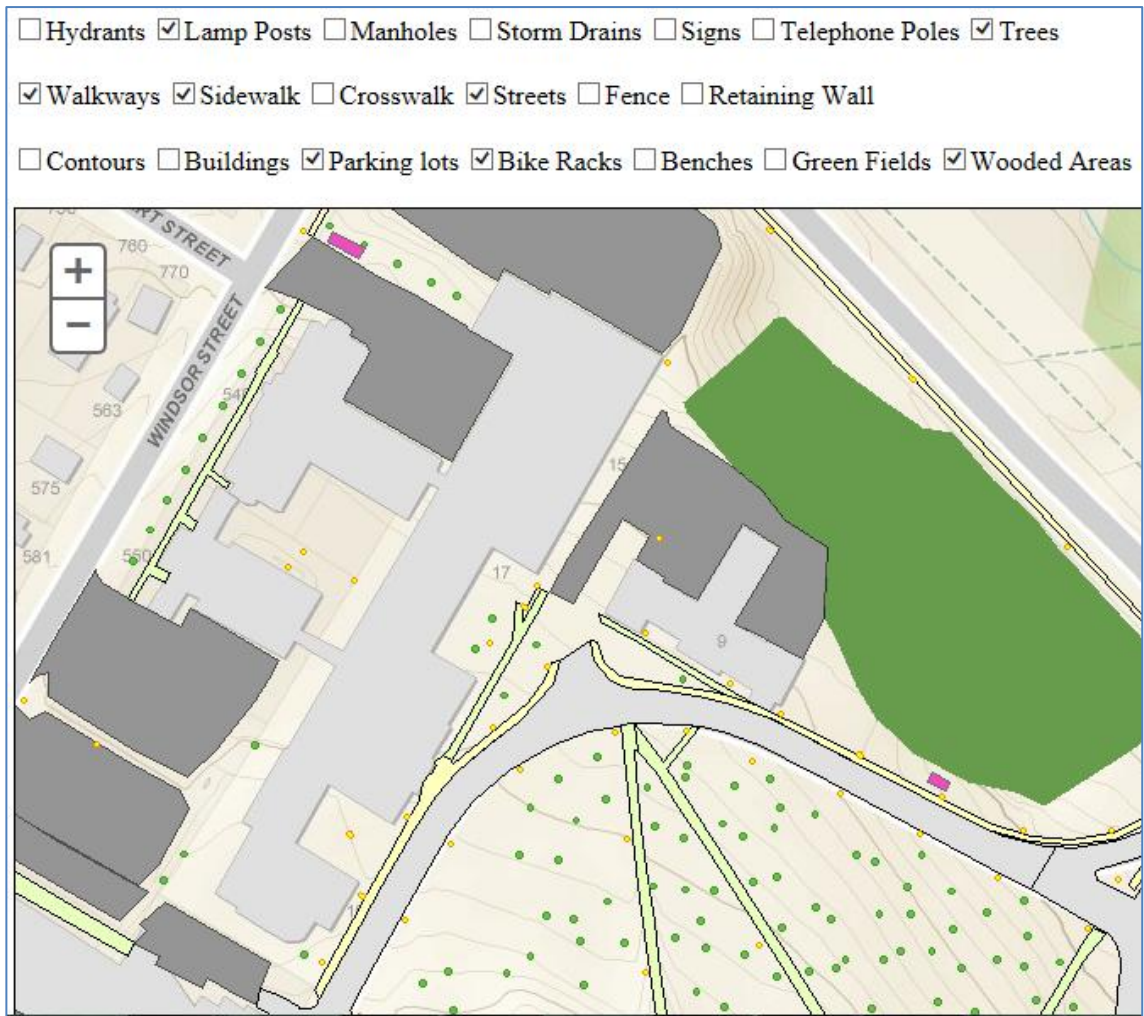


Figure III.2: Example of selected layers to display on the map service

It is important to know that point features (e.g. trees, lampposts, hydrants, manhole, etc.) are displayed in higher zooming levels than line and polygon features (e.g. parking lots, streets, green fields, etc.). While displaying the layers, clicking on the features will return a pop-up window containing the attribute data including the feature code, type, and spatial forms that were specified for those in the main database.

Examples have been given below for the features that have been selected on the map to retrieve their attribute values. Figure II.3 shows the selected building features along with the pop-up window showing the *Code: BL*, to use it for coding the survey data; *Feature Type: Building*, to describe the feature; and *Data Type in the Repository: Polygon*; *Polygon*, to define the spatial form of the feature to survey and then map in the CAD file.

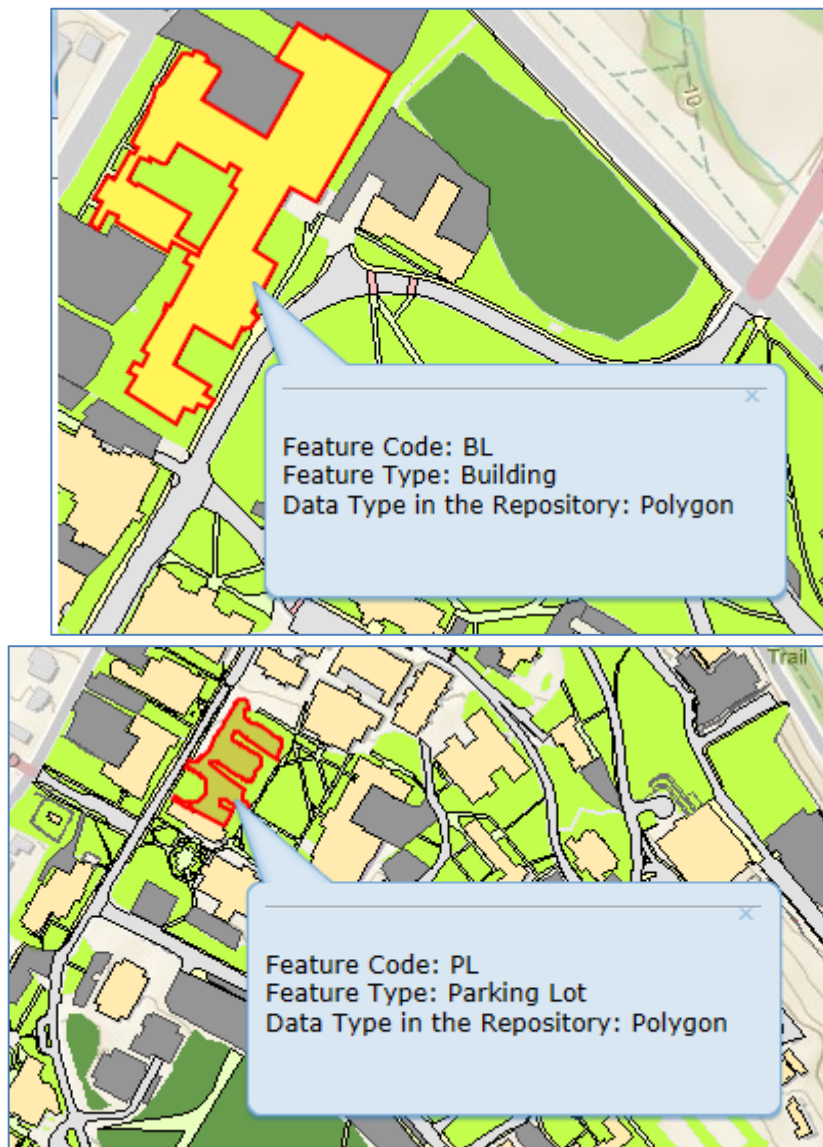


Figure III.3: Example of pop-up window indicating features attribute data

III.2 ArcCatalog Manager

The map Service was established in the ArcCatalog as mentioned in the steps below:

1. **Adding a new service** in the ArcCatalog. The map service was added and then the .MXD file defined as the input data for the service. The .MXD file includes the map layers that generated in the ArcMap and then saved as .MXD file.
2. **Defining the Service Type** to specify the capabilities of the service. In this case, the map service has the capability of map and query services. Finally, the map service established and then run in the ArcGIS server.
3. **Previewing the service** in the ArcCatalog to make sure if the service is active or not. Figure II.1 shows the map service in the ArcCatalog.

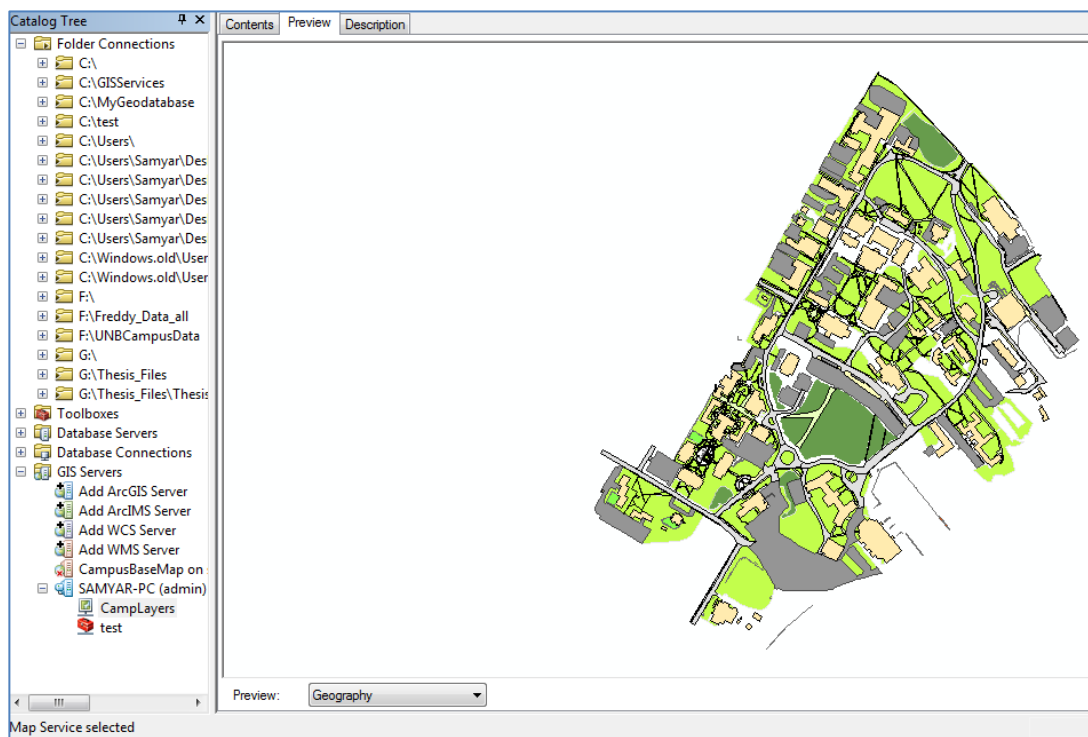


Figure III.4: Example of the map service preview in ArcCatalog

III.3 HTML and JavaScript Code

```
<!DOCTYPE html>

<html>

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

    <meta http-equiv="X-UA-Compatible" content="IE=7,IE=9">

    <!--The viewport meta tag is used to improve the presentation and behavior of the
samples
    on iOS devices-->

    <meta name="viewport" content="initial-scale=1, maximum-scale=1,user-
scalable=no">

    <title>SurveyCampLayer</title>

    <link rel="stylesheet"
href="http://serverapi.arcgisonline.com/jsapi/arcgis/3.3/js/dojo/dijit/themes/claro/claro.c
ss">

    <link rel="stylesheet"
href="http://serverapi.arcgisonline.com/jsapi/arcgis/3.3/js/esri/css/esri.css">

    <script>dojoConfig = { parseOnLoad:true };</script>

    <script src="http://serverapi.arcgisonline.com/jsapi/arcgis/3.3/"></script>

    <script>

      dojo.require("esri.map");

      dojo.require("esri.tasks.query");
```


//Defining the variables used in the functions and classes:

```
var layer, map;
```

```
var queryTask, queryTask1, queryTask2, queryTask3, queryTask4, queryTask5,  
queryTask6, queryTask7, queryTask8, queryTask9, queryTask10;
```

```
var queryTask11, queryTask12, queryTask13, queryTask14, queryTask15, queryTask16,  
queryTask17, queryTask18, queryTask19, queryTask20;
```

```
var query;
```

```
var visible = [];
```

```
var featureSet;
```

*//The main function which constructs the Basemap, map feature layer inputs, query
//inputs, and the query filters.*

```
function init() {
```

```
map = new esri.Map("map",{
```

```
basemap: "topo",
```

```
extent: new esri.geometry.Extent(-66.649725, 45.940313, -66.636766, 45.950831),
```

```
zoom: 16
```

```
});
```

*//Use the ImageParameters to set the visible layers in the map service during
//ArcGISDynamicMapServiceLayer construction.*

```
var imageParameters = new esri.layers.ImageParameters();
```

```
imageParameters.layerIds = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20];
```

```
imageParameters.layerOption = esri.layers.ImageParameters.LAYER_OPTION_SHOW;
```

```
//can also be: LAYER_OPTION_EXCLUDE, LAYER_OPTION_HIDE,
```

```
//LAYER_OPTION_INCLUDE
```

```
layer = new esri.layers.ArcGISDynamicMapServiceLayer("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer",
```

```
    {"imageParameters":imageParameters  
    });
```

```
map.addLayer(layer);
```

***//Listen for click event on the map, when the user clicks on the map call
//executeQueryTask function.***

```
    dojo.connect(map, "onClick", executeQueryTask);  
        dojo.connect(map, "onClick", executeQueryTask1);  
        dojo.connect(map, "onClick", executeQueryTask2);  
        dojo.connect(map, "onClick", executeQueryTask3);  
        dojo.connect(map, "onClick", executeQueryTask4);  
        dojo.connect(map, "onClick", executeQueryTask5);  
        dojo.connect(map, "onClick", executeQueryTask6);  
        dojo.connect(map, "onClick", executeQueryTask7);  
        dojo.connect(map, "onClick", executeQueryTask8);  
        dojo.connect(map, "onClick", executeQueryTask9);  
        dojo.connect(map, "onClick", executeQueryTask10);  
        dojo.connect(map, "onClick", executeQueryTask11);  
        dojo.connect(map, "onClick", executeQueryTask12);  
        dojo.connect(map, "onClick", executeQueryTask13);  
        dojo.connect(map, "onClick", executeQueryTask14);  
        dojo.connect(map, "onClick", executeQueryTask15);  
        dojo.connect(map, "onClick", executeQueryTask16);  
        dojo.connect(map, "onClick", executeQueryTask17);
```



```
dojo.connect(map, "onClick", executeQueryTask18);
dojo.connect(map, "onClick", executeQueryTask19);
dojo.connect(map, "onClick", executeQueryTask20);
```

//Listent for infoWindow onHide event

```
dojo.connect(map.infoWindow, "onHide", function() {map.graphics.clear();});
```

//Defining input data for the query task.

```
queryTask = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/0");

    queryTask1 = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/1");

    queryTask2 = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/2");

    queryTask3 = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/3");

    queryTask4 = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/4");

    queryTask5 = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/5");

    queryTask6 = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/6");

    queryTask7 = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/7");

    queryTask8 = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/8");

    queryTask9 = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/9");

    queryTask10 = new esri.tasks.QueryTask("http://samyar-
pc/arcgis/rest/services/CampLayers/MapServer/10");
```

```
        queryTask11 = new esri.tasks.QueryTask("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer/11");  
  
        queryTask12 = new esri.tasks.QueryTask("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer/12");  
  
        queryTask13 = new esri.tasks.QueryTask("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer/13");  
  
        queryTask14 = new esri.tasks.QueryTask("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer/14");  
  
        queryTask15 = new esri.tasks.QueryTask("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer/15");  
  
        queryTask16 = new esri.tasks.QueryTask("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer/16");  
  
        queryTask17 = new esri.tasks.QueryTask("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer/17");  
  
        queryTask18 = new esri.tasks.QueryTask("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer/18");  
  
        queryTask19 = new esri.tasks.QueryTask("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer/19");  
  
        queryTask20 = new esri.tasks.QueryTask("http://samyar-  
pc/arcgis/rest/services/CampLayers/MapServer/20");
```

//Defining the query filter

```
        query = new esri.tasks.Query();  
        query.outSpatialReference = {"wkid":4326};  
        query.returnGeometry = true;  
        query.outFields = ["FeatureCode","FeatureType","DataType"];  
  
    }
```

//Visibility Controller Function.

```
function updateLayerVisibility() {  
    var inputs = dojo.query(".list_item"), input;  
    //in this application layer 2 is always on.  
    visible = [];  
    for (var i=0, il=inputs.length; i<il; i++) {  
        if (inputs[i].checked) {  
            visible.push(inputs[i].id);  
        }  
    }  
    //if there aren't any layers visible set the array value to = -1  
    if(visible.length === 0){  
        visible.push(-1);  
    }  
  
    layer.setVisibleLayers(visible);  
}
```

// Query Task Execution Function

```
function executeQueryTask(evt) {  
    map.infoWindow.hide();  
    map.graphics.clear();  
    featureSet = null;  
  
    //onClick event returns the evt point where the user clicked on the map.  
    //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel  
    xy where the user clicked).
```

```

//set query geometry = to evt.mapPoint Geometry
    query.geometry = evt.mapPoint;

//Execute task and call showResults on completion
queryTask.execute(query, function(fset) {
    if (fset.features.length === 1) {
        showFeature(fset.features[0],evt);
    } else if (fset.features.length !== 0) {
        showFeatureSet(fset,evt);
    }
});
}

```

// Query Task-1 Execution Function.

```

function executeQueryTask1(evt) {
    map.infoWindow.hide();
    map.graphics.clear();
    featureSet = null;

    //onClick event returns the evt point where the user clicked on the map.

    //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
//xy where the user clicked).

    //set query geometry = to evt.mapPoint Geometry
    query.geometry = evt.mapPoint;

```

```
//Execute task and call showResults on completion
```

```
queryTask1.execute(query, function(fset) {  
  if (fset.features.length === 1) {  
    showFeature(fset.features[0],evt);  
  } else if (fset.features.length !== 0) {  
    showFeatureSet(fset,evt);  
  }  
});  
}
```

// Query Task-2 Execution Function

```
function executeQueryTask2(evt) {  
  map.infoWindow.hide();  
  map.graphics.clear();  
  featureSet = null;
```

```
//onClick event returns the evt point where the user clicked on the map.
```

```
//This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel  
xy where the user clicked).
```

```
//set query geometry = to evt.mapPoint Geometry
```

```
query.geometry = evt.mapPoint;
```

```
//Execute task and call showResults on completion
```

```
queryTask2.execute(query, function(fset) {  
  if (fset.features.length === 1) {  
    showFeature(fset.features[0],evt);  
  } else if (fset.features.length !== 0) {  
    showFeatureSet(fset,evt);  
  }  
});
```

```
}  
});  
}
```

// Query Task-3 Execution Function

```
function executeQueryTask3(evt) {  
    map.infoWindow.hide();  
    map.graphics.clear();  
    featureSet = null;  
  
    //onClick event returns the evt point where the user clicked on the map.  
    //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel  
    xy where the user clicked).  
    //set query geometry = to evt.mapPoint Geometry  
    query.geometry = evt.mapPoint;  
  
    //Execute task and call showResults on completion  
    queryTask3.execute(query, function(fset) {  
        if (fset.features.length === 1) {  
            showFeature(fset.features[0],evt);  
        } else if (fset.features.length !== 0) {  
            showFeatureSet(fset,evt);  
        }  
    });  
}
```

// Query Task-4 Execution Function

```
function executeQueryTask4(evt) {
    map.infoWindow.hide();
    map.graphics.clear();
    featureSet = null;

    //onClick event returns the evt point where the user clicked on the map.

    //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
    xy where the user clicked).

    //set query geometry = to evt.mapPoint Geometry
    query.geometry = evt.mapPoint;

    //Execute task and call showResults on completion
    queryTask4.execute(query, function(fset) {
        if (fset.features.length === 1) {
            showFeature(fset.features[0],evt);
        } else if (fset.features.length !== 0) {
            showFeatureSet(fset,evt);
        }
    });
}
```

// Query Task-5 Execution Function

```
function executeQueryTask5(evt) {  
    map.infoWindow.hide();  
    map.graphics.clear();  
    featureSet = null;  
  
    //onClick event returns the evt point where the user clicked on the map.  
    //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel  
xy where the user clicked).  
    //set query geometry = to evt.mapPoint Geometry  
    query.geometry = evt.mapPoint;  
  
    //Execute task and call showResults on completion  
    queryTask5.execute(query, function(fset) {  
        if (fset.features.length === 1) {  
            showFeature(fset.features[0],evt);  
        } else if (fset.features.length !== 0) {  
            showFeatureSet(fset,evt);  
        }  
    });  
}
```

// Query Task-6 Execution Function

```
function executeQueryTask6(evt) {  
    map.infoWindow.hide();  
    map.graphics.clear();  
    featureSet = null;
```



```

//onClick event returns the evt point where the user clicked on the map.

//This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
xy where the user clicked).

//set query geometry = to evt.mapPoint Geometry
query.geometry = evt.mapPoint;

//Execute task and call showResults on completion
queryTask6.execute(query, function(fset) {
    if (fset.features.length === 1) {
        showFeature(fset.features[0],evt);
    } else if (fset.features.length !== 0) {
        showFeatureSet(fset,evt);
    }
});
}

```

// Query Task-7 Execution Function

```

function executeQueryTask7(evt) {
    map.infoWindow.hide();
    map.graphics.clear();
    featureSet = null;

//onClick event returns the evt point where the user clicked on the map.

//This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
xy where the user clicked).

//set query geometry = to evt.mapPoint Geometry
query.geometry = evt.mapPoint;

```

```

//Execute task and call showResults on completion
queryTask7.execute(query, function(fset) {
  if (fset.features.length === 1) {
    showFeature(fset.features[0],evt);
  } else if (fset.features.length !== 0) {
    showFeatureSet(fset,evt);
  }
});
}

```

// Query Task-8 Execution Function

```

function executeQueryTask8(evt) {
  map.infoWindow.hide();
  map.graphics.clear();
  featureSet = null;

  //onClick event returns the evt point where the user clicked on the map.

  //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
  xy where the user clicked).

  //set query geometry = to evt.mapPoint Geometry
  query.geometry = evt.mapPoint;
  //Execute task and call showResults on completion
  queryTask8.execute(query, function(fset) {
    if (fset.features.length === 1) {
      showFeature(fset.features[0],evt);
    } else if (fset.features.length !== 0) {
      showFeatureSet(fset,evt);
    }
  });
}

```

```
    }  
  });  
}
```

// Query Task-9 Execution Function

```
function executeQueryTask9(evt) {  
  map.infoWindow.hide();  
  map.graphics.clear();  
  featureSet = null;  
  
  //onClick event returns the evt point where the user clicked on the map.  
  //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel  
  xy where the user clicked).  
  //set query geometry = to evt.mapPoint Geometry  
  query.geometry = evt.mapPoint;  
  
  //Execute task and call showResults on completion  
  queryTask9.execute(query, function(fset) {  
    if (fset.features.length === 1) {  
      showFeature(fset.features[0],evt);  
    } else if (fset.features.length !== 0) {  
      showFeatureSet(fset,evt);  
    }  
  });  
}
```

// Query Task-10 Execution Function

```
function executeQueryTask10(evt) {  
    map.infoWindow.hide();  
    map.graphics.clear();  
    featureSet = null;  
  
    //onClick event returns the evt point where the user clicked on the map.  
  
    //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel  
xy where the user clicked).  
  
    //set query geometry = to evt.mapPoint Geometry  
    query.geometry = evt.mapPoint;  
  
    //Execute task and call showResults on completion  
    queryTask10.execute(query, function(fset) {  
        if (fset.features.length === 1) {  
            showFeature(fset.features[0],evt);  
        } else if (fset.features.length !== 0) {  
            showFeatureSet(fset,evt);  
        }  
    });  
}
```

// Query Task-11 Execution Function

```
function executeQueryTask11(evt) {  
    map.infoWindow.hide();  
    map.graphics.clear();  
    featureSet = null;
```

```

//onClick event returns the evt point where the user clicked on the map.

//This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
xy where the user clicked).

//set query geometry = to evt.mapPoint Geometry
query.geometry = evt.mapPoint;

//Execute task and call showResults on completion
queryTask11.execute(query, function(fset) {
    if (fset.features.length === 1) {
        showFeature(fset.features[0],evt);
    } else if (fset.features.length !== 0) {
        showFeatureSet(fset,evt);
    }
});
}

```

// Query Task-12 Execution Function

```

function executeQueryTask12(evt) {
    map.infoWindow.hide();
    map.graphics.clear();
    featureSet = null;

//onClick event returns the evt point where the user clicked on the map.

//This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
xy where the user clicked).

//set query geometry = to evt.mapPoint Geometry
query.geometry = evt.mapPoint;

```

```

//Execute task and call showResults on completion
queryTask12.execute(query, function(fset) {
  if (fset.features.length === 1) {
    showFeature(fset.features[0],evt);
  } else if (fset.features.length !== 0) {
    showFeatureSet(fset,evt);
  }
});
}

```

// Query Task-13 Execution Function

```

function executeQueryTask13(evt) {
  map.infoWindow.hide();
  map.graphics.clear();
  featureSet = null;

  //onClick event returns the evt point where the user clicked on the map.

  //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
  xy where the user clicked).

  //set query geometry = to evt.mapPoint Geometry
  query.geometry = evt.mapPoint;
  //Execute task and call showResults on completion
  queryTask13.execute(query, function(fset) {
    if (fset.features.length === 1) {
      showFeature(fset.features[0],evt);
    } else if (fset.features.length !== 0) {
      showFeatureSet(fset,evt);
    }
  });
}

```

```
    }  
  });  
}
```

// Query Task-14 Execution Function

```
function executeQueryTask14(evt) {  
  map.infoWindow.hide();  
  map.graphics.clear();  
  featureSet = null;  
  
  //onClick event returns the evt point where the user clicked on the map.  
  //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel  
  xy where the user clicked).  
  //set query geometry = to evt.mapPoint Geometry  
  query.geometry = evt.mapPoint;  
  
  //Execute task and call showResults on completion  
  queryTask14.execute(query, function(fset) {  
    if (fset.features.length === 1) {  
      showFeature(fset.features[0],evt);  
    } else if (fset.features.length !== 0) {  
      showFeatureSet(fset,evt);  
    }  
  });  
}
```

// Query Task-15 Execution Function

```
function executeQueryTask15(evt) {  
    map.infoWindow.hide();  
    map.graphics.clear();  
    featureSet = null;  
  
    //onClick event returns the evt point where the user clicked on the map.  
  
    //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel  
xy where the user clicked).  
  
    //set query geometry = to evt.mapPoint Geometry  
    query.geometry = evt.mapPoint;  
  
    //Execute task and call showResults on completion  
    queryTask15.execute(query, function(fset) {  
        if (fset.features.length === 1) {  
            showFeature(fset.features[0],evt);  
        } else if (fset.features.length !== 0) {  
            showFeatureSet(fset,evt);  
        }  
    });  
}
```

// Query Task-16 Execution Function

```
function executeQueryTask16(evt) {  
    map.infoWindow.hide();  
    map.graphics.clear();  
    featureSet = null;
```



```

//onClick event returns the evt point where the user clicked on the map.

//This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
xy where the user clicked).

//set query geometry = to evt.mapPoint Geometry
query.geometry = evt.mapPoint;

//Execute task and call showResults on completion
queryTask16.execute(query, function(fset) {
    if (fset.features.length === 1) {
        showFeature(fset.features[0],evt);
    } else if (fset.features.length !== 0) {
        showFeatureSet(fset,evt);
    }
});
}

```

// Query Task-17 Execution Function

```

function executeQueryTask17(evt) {
    map.infoWindow.hide();
    map.graphics.clear();
    featureSet = null;

//onClick event returns the evt point where the user clicked on the map.

//This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
xy where the user clicked).

//set query geometry = to evt.mapPoint Geometry
query.geometry = evt.mapPoint;

```

```

//Execute task and call showResults on completion
queryTask17.execute(query, function(fset) {
  if (fset.features.length === 1) {
    showFeature(fset.features[0],evt);
  } else if (fset.features.length !== 0) {
    showFeatureSet(fset,evt);
  }
});
}

```

// Query Task-18 Execution Function

```

function executeQueryTask18(evt) {
  map.infoWindow.hide();
  map.graphics.clear();
  featureSet = null;

  //onClick event returns the evt point where the user clicked on the map.

  //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
  xy where the user clicked).

  //set query geometry = to evt.mapPoint Geometry
  query.geometry = evt.mapPoint;
  //Execute task and call showResults on completion
  queryTask18.execute(query, function(fset) {
    if (fset.features.length === 1) {
      showFeature(fset.features[0],evt);
    } else if (fset.features.length !== 0) {
      showFeatureSet(fset,evt);
    }
  });
}

```

```
    }  
  });  
}
```

// Query Task-19 Execution Function

```
function executeQueryTask19(evt) {  
  map.infoWindow.hide();  
  map.graphics.clear();  
  featureSet = null;  
  
  //onClick event returns the evt point where the user clicked on the map.  
  //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel  
  xy where the user clicked).  
  //set query geometry = to evt.mapPoint Geometry  
  query.geometry = evt.mapPoint;  
  
  //Execute task and call showResults on completion  
  queryTask19.execute(query, function(fset) {  
    if (fset.features.length === 1) {  
      showFeature(fset.features[0],evt);  
    } else if (fset.features.length !== 0) {  
      showFeatureSet(fset,evt);  
    }  
  });  
}
```

// Query Task-20 Execution Function

```
function executeQueryTask20(evt) {
    map.infoWindow.hide();
    map.graphics.clear();
    featureSet = null;

    //onClick event returns the evt point where the user clicked on the map.
    //This is contains the mapPoint (esri.geometry.point) and the screenPoint (pixel
    xy where the user clicked).

    //set query geometry = to evt.mapPoint Geometry
    query.geometry = evt.mapPoint;

    //Execute task and call showResults on completion
    queryTask20.execute(query, function(fset) {
        if (fset.features.length === 1) {
            showFeature(fset.features[0],evt);
        } else if (fset.features.length !== 0) {
            showFeatureSet(fset,evt);
        }
    });
}
```

//Show Selected Features for the Query

```
function showFeature(feature,evt) {
    map.graphics.clear();
```

```

//Set Symbol

var symbol = new
esri.symbol.SimpleFillSymbol(esri.symbol.SimpleFillSymbol.STYLE_SOLID, new
esri.symbol.SimpleLineSymbol(esri.symbol.SimpleLineSymbol.STYLE_SOLID, new
dojo.Color([255,0,0]), 2), new dojo.Color([255,255,0,0.5]));

feature.setSymbol(symbol);

//construct infowindow title and content

var attr = feature.attributes;

var title = attr.FIELD_NAME;

var content = "<br />Feature Code: " + attr.FeatureCode
            + "<br />Feature Type: " + attr.FeatureType
            + "<br />Data
Type in the Repository: " + attr.DataType;

map.graphics.add(feature);

map.infoWindow.setTitle(title);
map.infoWindow.setContent(content);
map.infoWindow.resize(250, 110);

(evt) ?
map.infoWindow.show(evt.screenPoint,map.getInfoWindowAnchor(evt.screenPoint)) :
null;
}

function showFeatureSet(fset,evt) {
//remove all graphics on the maps graphics layer
map.graphics.clear();

var screenPoint = evt.screenPoint;

```

```

featureSet = fset;

var numFeatures = featureSet.features.length;

//QueryTask returns a featureSet. Loop through features in the featureSet and
add them to the infowindow.

var title = "You have selected " + numFeatures + " fields.";
var content = "Please select desired field from the list below.<br />";

for (var i=0; i<numFeatures; i++) {
    var graphic = featureSet.features[i];
    content = content + " Field (<A href='#'
onclick='showFeature(featureSet.features[" + i + "]);'>show</A><br/>";
}

map.infoWindow.setTitle(title);
map.infoWindow.setContent(content);

map.infoWindow.show(screenPoint,map.getInfoWindowAnchor(evt.screenPoint));
}

dojo.ready(init);
</script>
</head>

```

<body>

This map service shows the survey camp feature layers and the feature coding system.

Camp Feature Layers:

 <input type='checkbox' class='list_item' id='0' value=0
onclick='updateLayerVisibility();'/>Hydrants

<input type='checkbox' class='list_item' id='1' value=1
onclick='updateLayerVisibility();'/>Lamp Posts

<input type='checkbox' class='list_item' id='2' value=2
onclick='updateLayerVisibility();'/>Manholes

<input type='checkbox' class='list_item' id='3' value=3
onclick='updateLayerVisibility();'/>Storm Drains

<input type='checkbox' class='list_item' id='4' value=4
onclick='updateLayerVisibility();'/>Signs

<input type='checkbox' class='list_item' id='5' value=5
onclick='updateLayerVisibility();'/>Telephone Poles

<input type='checkbox' class='list_item' id='6' value=6
onclick='updateLayerVisibility();'/>Trees

<input type='checkbox' class='list_item' id='7' value=7
onclick='updateLayerVisibility();'/>Walkways

<input type='checkbox' class='list_item' id='8' value=8
onclick='updateLayerVisibility();'/>Sidewalk

<input type='checkbox' class='list_item' id='9' value=9
onclick='updateLayerVisibility();'/>Crosswalk

<input type='checkbox' class='list_item' id='10' value=10
onclick='updateLayerVisibility();'/>Streets

<input type='checkbox' class='list_item' id='11' value=11
onclick='updateLayerVisibility();'/>Fence

```
<input type='checkbox' class='list_item' id='12' value=12
onclick='updateLayerVisibility();'/>Retaining Wall<br />
```

```
<br />
```

```
<input type='checkbox' class='list_item' id='13' value=13
onclick='updateLayerVisibility();'/>Contours
```

```
<input type='checkbox' class='list_item' id='14' value=14
onclick='updateLayerVisibility();'/>Buildings
```

```
<input type='checkbox' class='list_item' id='15' value=15
onclick='updateLayerVisibility();'/>Parking lots
```

```
<input type='checkbox' class='list_item' id='16' value=16
onclick='updateLayerVisibility();'/>Bike Racks
```

```
<input type='checkbox' class='list_item' id='17' value=17
onclick='updateLayerVisibility();'/>Benches
```

```
<input type='checkbox' class='list_item' id='18' value=18
onclick='updateLayerVisibility();'/>Green Fields
```

```
<input type='checkbox' class='list_item' id='19' value=19
onclick='updateLayerVisibility();'/>Wooded Area
```

```
<input type='checkbox' class='list_item' id='20' value=20
onclick='updateLayerVisibility();'/>Playgrounds
```

```
</span><br />
```

```
<br />
```

```
<div id="map" class="claro" style="width:100%; height:800px; border:1px solid
#000;"></div>
```

```
</body>
```

```
</html>
```


VITA

Candidates' full Name: Samyar Sepehr

University Attended: Islamic Azad University, B.E, 2009

Publication:

Sepehr, S. , D. Fraser and E. Stefanakis (2013). "A Geospatial Reference Framework for Survey Camps". Proceedings of the 26th International Cartographic Conference, Dresden, Germany, August