Integrated XML and GML in Geographical Information System

By

Mardiana Binti Abdul Rahman

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information System)

JUNE 2004

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Integrated XML and GML in Geographical Information System

by

Mardiana Binti Abdul Rahman

A project dissertation submitted to the
Information System Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION SYSTEM)

Approved by,

(Justin Dinesh Devaraj)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
JUNE 2004

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

MARDIANA BINTI ABDUL RAHMAN

ABSTRACT

This project basically concentrated on the study of eXtensible Markup Language (XML) and Geography Markup Language (GML) in Geographical Information System (GIS). The objective of the project is to convert the spatial data (e.g.: coordinates, area, etc) by using the XML and GML and then coding will be integrated and viewed in the web browser by using the Scalable Vector Graphic (SVG) technology. Basically, this project is done to find a new way to overcome the weaknesses of map digitizing and taking advantage of the GML technology in Geographical Information System. The project scope is concentrate on the usage of XML and GML in GIS. Research is done on XML technologies, which are provided for GML. The technologies included technology for encoding and data modeling (Data Type Definition, XML Schema), technology for transforming (XSLT) and technology for graphic rendering (SVG). Research on GML is focused on manipulation of spatial data to convert to simple features such as point, line and polygon. This project combines XML, GML and SVG technologies in order to meet the project objectives. In completing this project, waterfall model is use as the methodology for the system development. The project is developed according to the four phases of system development, which are planning, analysis, design and implementation. The discussion of this project will be more on GML compatibility and the advantages of using SVG to view the map. The simple display of map created will be able to show that GML is suits for handling geo-spatial data over the Internet. The user would be able to view the map and zooming feature is provided by SVG.

ACKNOWLEDGEMENT

Bismillah ar-Rahmani Ar-Raheem

In the Name of Allah, The Most Compassionate, the Most Merciful

First and foremost I would like to recite my greatest gratitude to the Most Merciful Allah for giving me the opportunity in completing this manuscript on time and without much hassle or problem. Without His observance in giving me the chance in finishing the report, there might be major problem which can resulted in delay of turning in the report in the time constrain.

In completing this preliminary report, there are some people that had been the backbone of the activities done in the complete of this text. I wouldn't have been able to finish up without their assistance, encouragement, and support either in terms of material, or spiritual. With this I would like to put some credit to them who has helped me through this time duration. They are as listed as beneath:

1. **Mr. Justin Dinesh Devaraj** – my supervisor (for giving me the guidelines and ways in producing a good output and full support in terms of knowledge input along this project)

2. **The *Backbone* Of This FYP Committee** – Ms Vivian, Mr. Shuib, and all IT/IS lecturers, (for giving full commitment in term of providing info about the final year project)

3. **Universiti Teknologi PETRONAS** – all UTP staff (for the full cooperation and providing me very convenient places to complete the project with the provided utilities)

4. **Parents and families** (for giving the full moral support in completing the report in addition to the consultation they have given during my troubled times)

5. **Friends** (as they have been there for me during the good and bad times as we stay together under the same varsity)

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND NOMENCLATURES

CRS	:	Coordinate Reference System
CT	:	Coordinate Transformation
ESRI	:	Environmental Systems Research Institute Inc.
GIS	:	Geographical Information System
GML	:	Geography Markup Language
GUI	:	Graphical User Interface
HTML	:	Hypertext Markup Language
WMS	:	Web Map Server
WFS	:	Web Feature Server
OGC	:	OpenGIS Consortium
SVG	:	Scalable Vector Graphics
SDLC	:	System Development Life Cycle
SGML	:	Standard Generalized Markup Language
UML	:	Unified Modeling Language
USGS	:	U.S. Geological Survey
UTP	:	Universiti Teknologi PETRONAS
VML	:	Vector Markup Language
XML	:	eXtensible Markup Language
XSD	:	XML Schema Definitions
XSL	:	eXtensible Stylesheet Language
XSLT	:	eXtensible Stylesheet Language
X3D	:	eXtensible 3D graphics

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

Chapter 1 explains the fundamental information of the project, which consists of background of study, problem statement, objective and scope of the project. A brief explanation of GIS also included in this section. This report basically gives details information about the GML history and its current development. The roles of XML and GML for the project are stated in this section since XML and GML are the main tools to accomplish the project.

1.1 Background of Study

Geographical Information System (GIS) is a computer based information system used to digitally represent and analyse the geographic features present on the Earth' surface and the events that taking place on it. GIS is also defined as "a computer system capable of capturing, storing, analyzing, and displaying geographically referenced information; that is, data identified according to location" [1].

A large part of Internet progress depends on the suitable approaches of dealing with data and information, and Markup Languages (MLs) play the main role in this progress. A Markup Language is a way of describing a document by placing tags in the document. Markup Languages vary from other programming languages, in containing loops, conditional logics, subroutines and some other programming structures. There are many Markup Languages with different applications these days such as HTML, XML, SGML, and VML.

The eXtensible Mark-up Language (XML) has been developed to compliment HTML, which uses the same tag structure as HTML. With XML, the element can declare its associated data to be an address or posts code, a point value with associated attribute data or any other desired data element. As in GIS area, the use of XML will make users able to search and manipulate on-line data, regardless of derived application. Retrieval of such data can allow for manipulation in spatially enabled web browsers or local XML-aware GIS applications.

GML is a markup language that is based on the XML standard to construct structured spatial and non-spatial information to enable data sharing and interchange over the Web. On June 13, 2000, OpenGIS Consortium has published the first public release of a recommendation defining the Geography Markup Language (GML), Version 1.0. The Geography Markup Language (GML) is defined as an XML encoding for the transport and storage of geographic information, including both the geometry and properties of geographic features [3]. GML will make a significant impact on the ability of organizations to share geographic information with one another, and to enable linked geographic datasets. This specification defines the mechanisms and syntax that GML uses to encode geographic information in XML.

GML has become the common base for many applications and give many advantages. Before this, user needs to have a heavyweight desktop GIS in order to do GIS work. But this has been reduced since GML provided standardized way of graphic features and a new way of dealing geographic information via the Internet. For example visualization does not require a GIS tool at all, because a simple transformation of the GML to another XML based format like Scalable Vector Graphics (SVG) or eXtensible 3D graphics (X3D) can be accomplished through some scripting and a XSL (eXtensible Stylesheet Language) file describing how to represent the GML features and a commonly available transformation engine which most platforms have built in today [2]. Since GML is best represent the geographic data content, so GML also suitable for making maps.

1.2 Problem Statement

For developing web pages there are ever-expanding numbers of hypertext mark-up language (HTML) tags that can be used to format the way items look on pages. At the same time, it might be possible to define a hyperlink that can be used to open a spatially-enabled application, or to link to an internet mapping system, which can be time consuming to implement and difficult to extend. As today's web browsers excel at viewing data on the Internet, there is a growing need to evolve beyond this.

When reviewing to the past years, user needs to digitize the map in order to get the desired data from the maps. Digitizing is one of the tools to convert spatial data to digital format and one of the oldest techniques of digital input. Digitizing cause a problem since maps are not designed to be digitized. This technique also relies on the skill of operators entering data and usually deals with the overshoot problem, and incorrect data input.

This project is basically focusing on manipulating the data (coordinates) by using XML and GML technology. XML and GML will be use to plot up the data to point, line and polygon object and then the code will be transformed into SVG in order to visualize it.

1.3 Objectives The Project

This project is done in two phases. In the first phase, the research has been done to understand the concept of XML and GML in GIS. In the second phase, the design and development of the map by implementing the XML and GML are to be carried out. Objectives of the project to be achieved are as follow:

i. To understand the concepts of XML and GML in GIS.
ii. To do a research on both technologies, which are XML and GML in GIS area.
iii. To create a simple display of map by converting the spatial data (e.g.: coordinates, area, etc) using the XML and GML and view it in the web browser.

1.4 Scope Of Study

The research area for this project will cover the Geographic Information System regarding the use of XML and GML. Research is done on XML technologies, which are provided for GML. The technologies included technology for encoding and data modeling (Data Type Definition, XML Schema), technology for transforming (XSLT) and technology for graphic rendering (SVG). Research on GML is focused on manipulation of spatial data to convert to simple features such as point, line and polygon. The data (coordinates) is obtained from the Internet. The data such as coordinates for the building will be plotting using XML and GML. The code then will be transform into SVG by using the XSLT. Then it will be visualized in the web browser (Internet Explorer). Basically, this project is to approve that the data (coordinate) can be plot up to build the point, line, and polygon object by using XML and GML. This project focused on developing simple display of map (point, line or polygon) in a web browser using SVG.

CHAPTER 2

LITERATURE REVIEW AND THEORY

2. LITERATURE REVIEW AND THEORY

This chapter contains a more focused scope on XML and GML in Geography Information System. It provides the information about how XML and GML is implementing over the Internet. This chapter contains the acknowledged findings on this field, consisting of relevant theories, hypothesis, facts and data, which are relevant to the objective and the research of this project. An overview of GIS, its components and software also is introduced in this chapter. More elaborations are given on XML, GML, SVG and the open source for GIS.

2.1 Geography Information System (GIS)

A GIS is a computer system capable of capturing, storing, analyzing, and displaying geographically referenced information; that is, data identified according to location [1]. The power of a GIS comes from the ability to relate different information in a spatial context and to reach a conclusion about this relationship.

GIS stores descriptive information about features along with the spatial data. The descriptive data can be used to make maps, which show how a phenomenon such as elevation, soil type, or temperature varies from place to place. Usually, the data is store in the form of layers connected by a common geographical frame of reference where each layer holds data about a particular kind of feature.

Map is one of the most common products of GIS. By using GIS, maps are commonly easy to make. Maps are often the most effective means of conveying the results of the GIS process. Therefore, GIS is usually a productive producer of maps. Data capture is putting the information into the system that involves identifying the objects on the map, their absolute location on the Earth's surface, and their spatial relationships. Software tools that automatically extract features from satellite images or aerial photographs are gradually replacing what has traditionally been a time-consuming capture process. Objects are identified in a series of attribute tables, which is the information part of a GIS. Spatial relationships, such as whether features intersect or whether they are adjacent, are the key to all GIS-based analysis.

2.1.1 GIS Components

GIS relies on five important components, which are hardware, software, data, system, and people. GIS relies on a computer for storage and processing of data. The size of the computing system will depend on the type and nature of the GIS. A small scale GIS will only need a small personal computer to run on, while a large enterprise wide system with larger computers and a host of client machines to support multiple users. Data for GIS comes in two forms geographic or spatial data, and attribute or spatial data. Spatial data are data that contain an explicit geographic location in the form of a set of coordinates. Attribute data are descriptive sets of data that contain various information relevant to a particular location.

GIS systems are designed and developed to aid the data management and decision support processes of an organization. The operation of any organization is based on a set of practices and business logic unique to that organization. While some organizations may use a GIS on an ad-hoc basis with each user formulating their own standards of work and methods of analysis others define their business logic into the GIS to streamline certain aspects of their operations. GIS system lies the GIS software itself providing the functionality to store, manage, link, query and analyze geographic data. The system users - those who will use the GIS to solve spatial problems - are most often people who are well trained in GIS, perhaps in a specific application of GIS. System operators are responsible for the day-to-day

operations of the system, more often performing tasks that allow the system users to function efficiently.

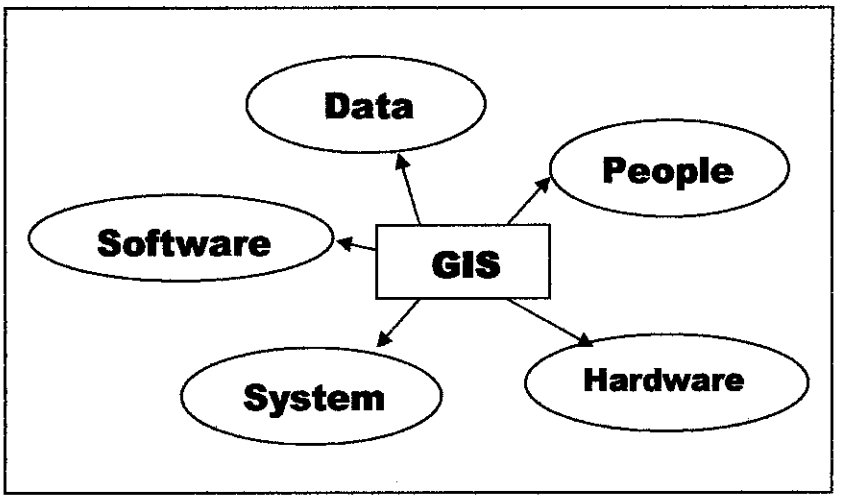


Figure 2.1: The GIS components

2.1.2 GIS Data Models

In GIS, all graphical features on the earth can be represented by only three identities that are line, point and polygon. The layers of data are stored in the GIS using one of two distinctly different data models, known as raster and vector. In raster model, a feature is defined as a set of cells on a grid. All of the cells on the grid are of the same shape and size and each one is identified by a coordinate location and a value which acts as its identifier, features are represented by a cells or groups of cells that share the same identifier. Raster data are applied in at least four ways, which are models describing the real world, digital image scans of existing maps, compiling digital satellite and image data and automatic drawing driven by raster output units. The raster model is particularly useful for working with continuous forms of features such as soil types, vegetation etc.

In vector, a feature is represented as a collection of begin and end points used to define a set of points, lines or polygons which describe the shape and size of the feature. The vector model is particularly useful for representing highly discrete data

types such as roads, buildings, boundaries and the lake. A point can be used to represent an area such as a city, or an earthquake epicenter. The line is used to portray linear features such as highways, and earthquake fault zones. If several lines are joined into a closed figure, a polygon is created. National boundaries, vegetation, or geologic formations are typically mapped as polygons. Vector GIS can store corresponding information of complex objects more efficiently. In this project, the data model will be more focused on vector data model.

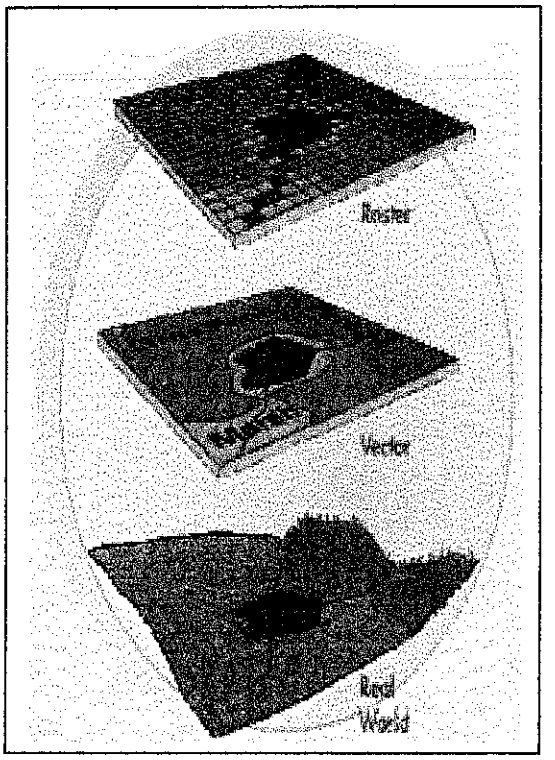


Figure 2.2: GIS Data Models

2.1.3 GIS Data Store

GIS stores a representation of the world in the form of layers connected by a common geographical frame of reference. Each of the features on a layer has a unique identifier, which distinguishes it from the rest of the features on the layer and allows you to relate it to relevant information stored in external databases. This simple yet powerful mode of abstraction, a GIS allows us to capture only those elements of the world that are of interest to us. Different views and data about the

world e.g., streets, soils, pipes, cables, vegetation, etc. can be captured and stored in the GIS over time to accommodate the needs of various different users and to reflect changes in the landscape over time.

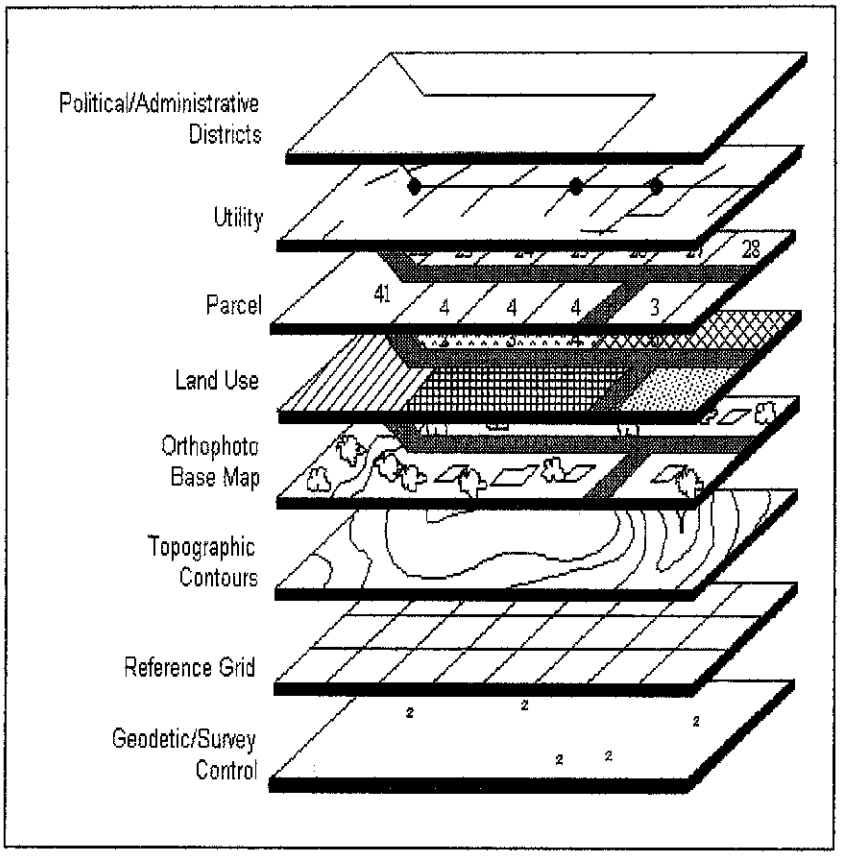


Figure 2.3: GIS Data Store

2.1.4 Geospatial Data

Geospatial data has both spatial and thematic components. Spatial information is presented in two ways: as vector data in the form of points, lines, and areas or polygons; or as grid data in the form of uniform, systematically organized cells. Conceptually, geographic data can be broken up in two elements, observation or entity and attribute or variable. GIS is able to manage both elements. Spatial component observations have two aspects in its localization: absolute localization based in a coordinates system and topological relationship referred to other

observations. A GIS is able to manage both while computer assisted cartography packages only manage the absolute one. Thematic component is considered as the variables or attributes studied considering the thematic aspect (statistics), the locational aspect (spatial analysis) or both (GIS).

2.1.5 GIS Software

GIS software typically includes tools for creating and editing spatial data such as from a database of latitude/longitude coordinates or by tracing aerial photographs or paper maps. It also includes tools for measuring distances and calculating the area and perimeter of features.

GIS software provides the functions and tools needed to store, analyze, and display information about places. There are four key components of GIS software which are tools for entering and manipulating geographic information such as addresses or political boundaries, a database management system (DBMS), tools that create intelligent digital maps which can analyze, query for more information, or print for presentation and an easy-to-use graphical user interface (GUI).

GIS software ranges from low-end business-mapping software appropriate for displaying sales territories to high-end software capable of managing and studying large protected natural areas. [5]

2.1.5.1 Arcview3.2/8

ArcView is software that can be used to display and print data and to carry out analysis. ArcView also has some limited data creation and editing capabilities. ArcView can import data from a limited number of GIS data formats. Arcview has no features of GML data import. This software is produced by Environmental Systems Research Institute, Inc. (ESRI).

2.1.5.2 ArcGIS

ArcGIS is a family of software products that form a complete GIS for geographic data creation, management, integration, and analysis. This software is produced by Environmental Systems Research Institute, Inc. (ESRI).

2.1.5.3 MapInfo Professional v7.0

MapInfo Professional is the GIS leading business mapping solution, which lets user to perform sophisticated and detailed data analysis to increase revenue, lower costs, boost efficiency and improve service with location-based intelligence. It is use to create highly detailed maps to enhance presentations and aid in decision making and reveal patterns and trends in data that may otherwise be impossible to see. Sophisticated and extensive data analysis also can be performed by using MapInfo. It also can import data from a large number of GIS data formats included GML.

2.2 eXtensible Markup Language (XML)

XML is a markup language, which is an enhancement of the HTML. XML stands for eXtensible Markup Language and it was designed to describe data or to create structured documents. It is basically created to structure, store and to send information. It uses the same tag structure as HTML. It also allows everyone to create his own information, send anything to anywhere for anybody. This means that it only focus on data and what data is without having the predefined defining tags and data. User may define the tags according to their aims and preference [7]. XML also has the capability of SGML and simplicity of HTML. XML also is used to create other markup languages for particular applications such as GML and VML. XML document has three main components, which is the structure of XML document, DTD or schema, and XSLT. These components are briefly explained below.

2.2.1 XML Document

XML document includes of a prolog and elements. The prolog for an XML document states some information to the parsers. This information expresses that the document is marked up in XML and can contain XML processor instruction. The prolog also includes text encoding, declaration of special pieces of text, and the DTD or schema being used [7]. The elements come after the prolog and XML tags will be define here.

```
<?xml version="1.0"?>
  <!DOCTYPE statement SYSTEM "fyp.dtd">
  <statement>
          <title>Final Year Project </title>
            ...
   </statement>
```

Figure 2.4: General Structure of a XML Document

2.2.2 Data Type Definition (DTD)

DTD can be defined as the grammar of the XML page. It actually is a tool to create and describe XML tags. Once a DTD is created and a document has been written based on that DTD, the document will be compared to the DTD that will cause the validation of the document. If the XML document follows the rules listed in the DTD, then the document is said to be valid, otherwise it is called invalid.

```
<?xml version="1.0"?>
<!DOCTYPE statement [
  <!ELEMENT statement (title,.....)>
  <!ELEMENT title     (#PCDATA)>
  .......
   ]>
```

Figure 2.5: General Structure of a DTD Document

2.2.3 XML Schema

Schema has the capability of DTD but differs in some characteristics; such as it is predefined for a specific application. In general terms a schema defines the characteristics of a class of objects; in XML a schema also describes how data is marked up. In this project, XML Schema standard is used to describe GML definitions. The rules and format to use for encoding the spatial features, such as points, lines and polygons in GML is stated in the XML Schema files. By using XML schema, GML also can be use for data modeling besides being as encoding language for geographic information. This schema also fits within an object-oriented framework.

GML schema documents are XML Schemas that define XML types and XML elements to encode GML objects with identity, elements to encode GML properties of those objects, and XML attributes qualifying those properties [3]. Usually the selection of schema depends on the features that we want to build. As example if we are modeling geographic features than we will need the feature.xsd, which is provided by the OGC.

The Unified Modeling Language (UML) offers a fairly general means of visually representing the elements of an application schema; a class diagram presents a concise overview of defined types, and a package diagram depicts higher-level groupings of model elements. UML representation of the geometry schema and feature schema are shown in Appendix 5 and Appendix 6.

XML Schema offers several advantages when it comes to constraining GML encodings such as it enables the intermingling of different vocabularies using namespaces, it permits finer control over the structure of the type definition hierarchy; and it confers extensibility and flexibility via derived types and substitution groups.

2.2.2 Extensible Stylesheet Language Transformations (XSLT)

A natural programming language for performing transformation is Extensible Stylesheet Language Transformations (XSLT). XSLT has become a standard means of transforming XML documents. Even beyond data transformation, XSLT is robust enough as a processing language to implement complete application logic for most web-based software requirements. In this project, SVG graphics are written in XSLT in order to transform the GML data to a SVG map since XSLT is a powerful and highly flexible tool for transforming GML data into SVG.

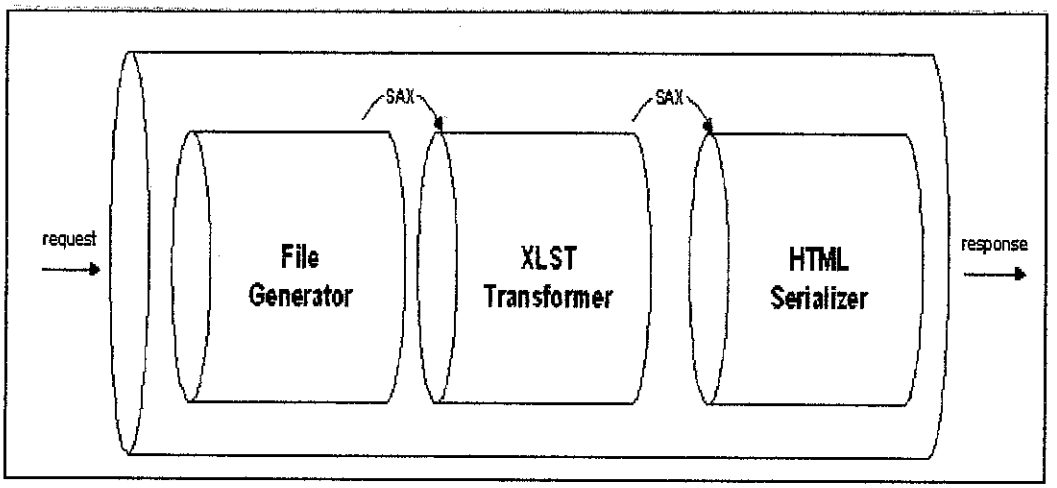


Figure 2.6: Overview of the XSLT with pipeline technology

2.3 Geographic Markup Language (GML)

Traditionally, geographic information has been developed and used by geographic community only. Increasing technology in today's world make geographic information available via the Internet. Currently, there is a new language which known as Geography Markup Language or GML which created for handling geo-spatial data over the Internet. Geography Markup Language (GML) is a standardized means of storing geographic information in eXtensible Markup Language (XML) encoded files specified by the openGIS Consortium (OGC). XML, an open, ASCII based, format uses descriptive tags to store data doing away with any proprietary vendor specific formats. Tags may be nested within each other and may be extended

in an object oriented like manner to suit your own data model, while maintaining compatibility with the standard [2].

GML as a kind of Markup Languages is a new way to deal with geographic information via the Internet. It is an XML encoding for the transport and storage of geographic information, including both the spatial and non-spatial properties of geographic features [3]. GML is a very useful and is a simple markup language for GIS applications that is extended by XML. GML defines the various entities such as features, geometries, topologies etc. through a hierarchy of GML objects as shown in the UML diagram in Appendix 4.

GML provides and XML-based encoding of geo-spatial data; it can be viewed as a basic application framework for handling geographic information in an open and non- proprietary way. By leveraging related XML technologies, a GML dataset becomes easier to process in heterogeneous environments, and it can be readily intermixed with other types of data: text, video, imagery, etc. Since GML documents are both human – readable and machine parsable, they are easier to understand and maintain than proprietary binary formats [13].

GML is suited for distribution over the network since the files may be streamed so that user does not have to wait for downloading an entire file before opening. With this advantage, it allow usability enhancement in a network environment. The interpretation of the GML content usually involves using graphical symbols, lines styles, and area or volume fills, and often some sort of transformation of the geometry of the GML data into the geometry of the visual presentation [4]. Figure 2.7 shows the general structure of GML document.

Since GML is an XML application, it can be readily styled into a variety of presentation formats including vector and raster graphics, text, sound and voice. Generation of graphical output such as maps is one of the most common presentations of GML. This presentation can be accomplished in a variety of ways including direct rendering by graphical applets or styling into an XML graphics technology such as SVG or eXtensible 3D Graphic (X3D). By using GML, the map that contains geographic information can be distributed over the Internet.

```
<pex:House>
        <pex:noRooms>8</ pex:noRooms >
        <pex:sellingPrice>120000</pex:sellingPrice>
        <pex:floorArea>2800</floorArea>
        <gml:extentOf>
                <gml:Polygon srsName = " ... " />
                    <gml:outerBoundaryIs>
                        <gml:LinearRing>
                            <gml:coordinates>.... </gml:coordinates>
                        </gml:LinearRing>
                    </gml:outerBoundaryIs>
                </gml:Polygon>
        </gml:extentOf>                    .....
</ pex:House >
```

Figure 2.7 General Structure of a GML Document

2.3.1 GML Conceptual Framework

GML provides a variety of kinds of objects for describing geography including features, coordinate reference systems, geometry, topology, time, units of measure and generalized values.

A geographic feature is an abstraction of a real world phenomenon; it is a geographic feature if it is associated with a location relative to the Earth. So a digital representation of the real world can be thought of as a set of features. The state of a feature is defined by a set of properties, where each property can be thought of as a {name, type, value} triple.

The number of properties a feature may have, together with their names and types, are determined by its type definition. Geographic features with geometry are those with properties that may be geometry-valued. The formal and descriptive names for the basic geometric properties are listed in Table 2.1; these names appear in the Feature schema to designate common geometric properties. A feature collection is a collection of features that can it be regarded as a feature; as a consequence a feature collection has a feature type and thus may have distinct properties of its own, in addition to the features it contains.

Formal name	Descriptive name	Geometry type
boundedBy	-	Box
pointProperty	location, position, centerOf	Point
lineStringProperty	centerLineOf, edgeOf	LineString
polygonProperty	extentOf, coverage	Polygon
geometryProperty	-	*any*
multiPointProperty	multiLocation, multiPosition, multiCenterOf	MultiPoint
multiLineStringProperty	multiCenterLineOf, multiEdgeOf	MultiLineString
multiPolygonProperty	multiExtentOf, multiCoverage	MultiPolygon
multiGeometryProperty	-	MultiGeometry

Table 2.1 Basic Geometric Properties

2.3.2 GML Application Schema

GML is designed to support interoperability and does so through the provision of basic geometry tags (all systems that support GML use the same geometry tags), a common data model (features/properties), and a mechanism for creating and sharing application schemas. Most information communities will seek to enhance their interoperability by publishing their application schemas; interoperability may be further improved in some cases through the use of profiles.

From the application point of view, it simplifies and standardizes the operations in many sectors, from map building to data format transformation, from spatial query to geographical analysis, including the emerging applications in mobile systems [31]. Since the GML data structure is XML-compliant, it can be transformed in a SVG document format and then easily displayed on a standard web browser.

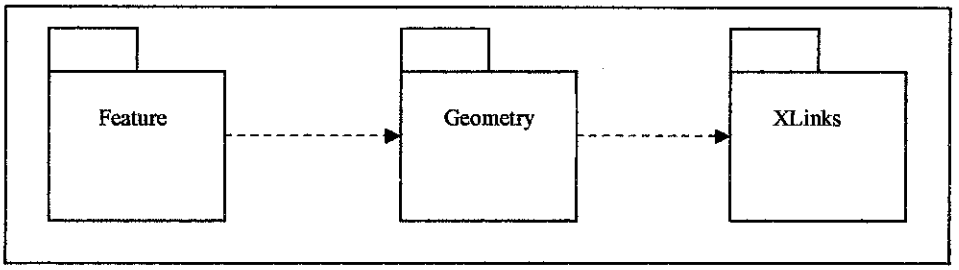


Figure 2.8: Base schemas as packages

2.4 Scalable Vector Graphic (SVG)

A main goal of GML is to represent the content of geographical data. This language can be used also to represent these data as maps, by using a rendering tool to interpret the GML data. In other words, it is necessary to "re- code" the GML elements in a suitable way so that they can be represented, for example, by the graphical display of a web browser (map styling). This operation interprets the GML contents by means graphical symbols, line styles, areas filling, sometime also the transformation of the data geometry according to the representation requested

Generally the graphical rendering process transforms the GML data in a XML graphical format. Some major vector graphical formats for viewing is:

i. Scalable Vector Graphics (SVG);
ii. Vector Markup Language (VML);
iii. Web 3D by X3D Consortium.

Today a variety of graphical render programs are available for the various XML graphical formats, both as native in the web browser, as plug-in for the browser, as stand-alone viewer, or as library of functions. Geography Markup Language (GML) depends on a viewing format such as SVG for rendering GML objects to the browser. SVG is an XML grammar for describing 2-D graphics, which includes elements for vector shape features, raster images, animation and text [8]. SVG represents a fundamental extension of the Internet, allowing vector design files to have full access to the Web.

This XML grammar, or tag language, can be processed with standard XML tools, such as validating parsers, editors and browsers. SVG is already supported by several browsers and provide by Adobe. In the map styling process, the SVG Explorer application transforms the GML elements in SVG elements in order to represent them in a graphical way. This operation assigns to each GML tag, referred to the geometrical property of a geographical element, a SVG tag and establishes a sort of correspondence between these tags.

SVG also integrates standard image formats, such as GIF, PNG and JPG. SVG vector features can be overlaid on raster images, making hybrid raster/vector displays possible. SVG is compatible with the full range of XML specifications that are becoming available to the Internet. XPath, XPointer, XQuery, XForm, XHTML and XSLT are a few of the XML technologies that provide basic infrastructure for a web. As part of this XML infrastructure, SVG stands firmly centered in the flow of current Internet technologies.

The SVG Explorer application is running on a web browser (Microsoft Explorer). The application's main goal is the processing of GML documents and their visualization in a graphical way, with the interaction of the user. A set of basic functionalities handling the graphical (zoom, pan, symbols, styles, etc), the geographical and the thematic aspects (multilayer organization, elements classification and aggregation by attributes) has been developed.

2.5 OpenGIS Consortium, Inc (OGC)

Open GIS Consortium, Inc or also known as OGC is an international industry consortium of 259 companies, government agencies and universities involve in a consensus development to produce publicly available interface specifications. OGC mission is to deliver spatial interface specifications that are openly available for global user, which supports interoperable solutions that "geo-enable" the Web, wireless and location-based, services, and mainstream IT. The specifications empower technology developers to make complex spatial information and services available and practical with all kinds of applications.

OpenGIS® is a Registered Trademark of the OGC and is the brand name associated with the Specifications and documents produced by the OGC. OpenGIS specifications are developed in a unique consensus process supported by OGC industry, government and academic members to enable geoprocessing technologies to interoperate, or "plug and play" [10].

The Open GIS Consortium Inc. (OGC), has recently approved and released Geography Markup Language version 3.0 (GML 3), which defines a data encoding in XML that allows geographic data and their attributes to be moved among disparate systems. The new release is modular, meaning that users can select out only the parts necessary for use, which simplifies and minimizes the size of implementations. New additions in GML 3 include support for complex geometries, spatial and temporal reference systems, topology, and units of measure, metadata, gridded data, and default styles for feature and coverage visualization [11]. GML 3 is almost entirely backwards compatible with GML 2, so developers and users who familiar with GML 2 can begin working immediately with GML 3.

CHAPTER 3

METHODOLOGY AND PROJECT WORK

3. METHODOLOGY AND PROJECT WORK

This chapter features the detailed description of methodology and procedure of completing this project. This methodology is implemented in order to ensure that the project is running as required. An overview of system development methodology is also described in this chapter. The methodology used for this project is waterfall model.

3.1 System Development Methodology

An information system development methodology is defined as a collection of procedures, techniques, tools, and documentation aids which will help the systems developers in their efforts to implement new information system. A methodology will consist of phases, themselves consisting of sub-phases, which will guide the systems developers in their choice of techniques that might be appropriate at each stage of the project and also help them plan, manage, control and evaluate information systems projects [16].

System Development Life Cycle (SDLC) refers to a methodology for developing systems. The SDLC methodology tracks a project from an idea developed by the user, through a feasibility study, systems analysis and design, programming, pilot testing, implementation, and post-implementation analysis. Documentation developed during the project development is used in the future when the system is reassessed for its continuation, modification, or deletion.

In this project, the methodology chosen was waterfall model. The decision was made after analyzing the advantages and disadvantages implementing the methodology in the project development. It provides a consistent framework of tasks and deliverables needed to develop the system.

3.2 Waterfall Model

Methodology plays a vital role in completing any project. Waterfall model is used as the methodology to plan and manage the system development process for this project. All the phases in the system development life cycle (SDLC) applies to this model in order to develop the project as shown in Figure 3.1. The waterfall model consist of 5 important phases that are:

a. Planning
b. Analysis
c. Design
d. Implementation & Testing
e. Operation & Support

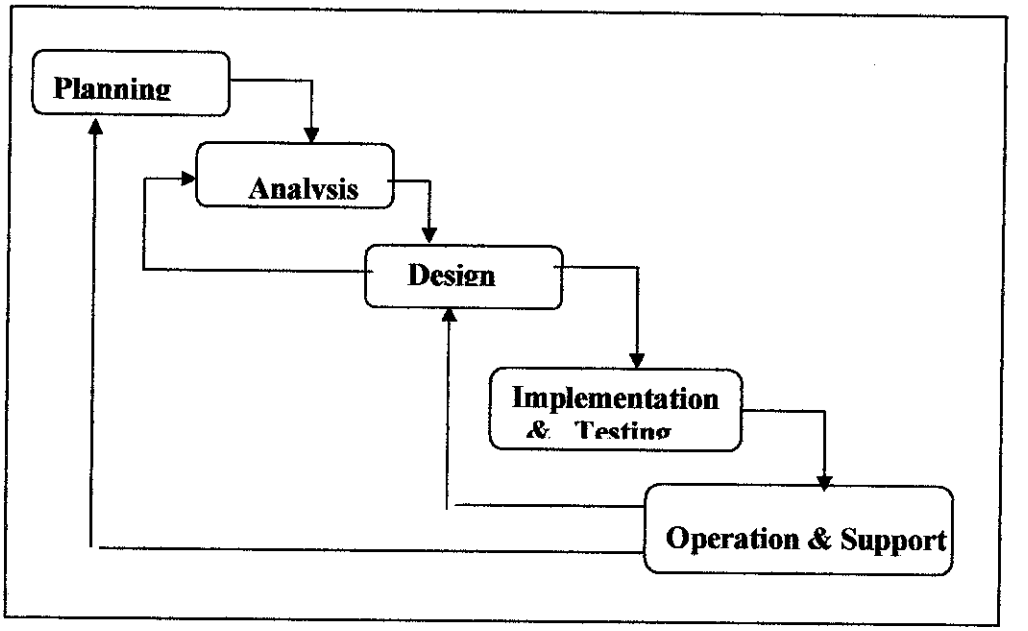


Figure 3.1: System Development Life Cycle for Waterfall Model

3.2.1 Planning

System planning begins with a formal proposal or request for the project. Proposal was developed and submitted to the FYP committee. Scope of study was also established during this period. In this phase, the purpose is to identify clearly the nature and scope of the business opportunity or problem by performing preliminary investigation or also called as feasibility study. This phase expands on the high-level project outline and provides a specific and detailed project definition. The outcome from this study is project scope. The project scope is concentrate on the study of Geographic Information System regarding the use of XML and GML.

This preliminary investigation is a critical step since the outcome will affect the entire of development process. The feasibility study is used to determine if the project should get the go-ahead. The study included the technical aspect for the project such as the hardware and software requirement, and also the possible sources or references for the project. OGC or OpenGIS Consortium has been identified as the main resources for this project since the consortium is the one who initiated GML. The project proceeded with producing a project plan and project schedule for the future stages of development.

3.2.2 Analysis

The purpose of this phase is to understand the requirements and build a logical model for the system. As implement in this project, this is the phase of doing research and analysis. Information, findings, relevant theories, hypothesis, facts and data, which are relevant to the objective and the scope of research for this project were collected as much as possible during this stage.

Researches on XML and GML were done during this stage. The integration between both markup language were identified and their pro and cons were discussed. Research is done on XML technologies, which are provided for GML. The

technologies included technology for encoding and data modeling (Data Type Definition, XML Schema), technology for transforming (XSLT) and technology for graphic rendering (SVG). Research on GML is focused on manipulation of spatial data to convert to simple features such as point, line and polygon.

Several projects regarding the topic also were analyzed in order to come out with a good product analysis. Galdos Inc was identified as the main company, which provided the GML technology and done many research on this technology. The company also provided list of tools needed for the project. The tools were analyze and reviewed against in order to check the availability status. Tools that are used for the project are XML editor; XMLSpy, XML parser; Xerces-J v1.3.1, XSLT processor; Saxon, and graphics rendering; Adobe SVG Viewer.

At this stage, the preliminary report was sent to the supervisor as required. It contains the project details such as the scope, project objectives and requirements of the project, project plan and analysis done for the project.

3.2.3 Design

In this phase, all necessary outputs, inputs, interfaces, and processes were identified. The tools needed for the design phase were downloaded and installed. Some of the tools are free download from Internet such as XML parser - Xerces, XMLSpy from Altova, Inc, Adobe SVG Viewer, and XSLT processor - Saxon. The development started with developing the GML documents.

All the XML Schemas needed were identified and download from the open source. The Unified Modeling Language (UML) was developed during this phase as it was done in order to model the GML document. The data used for this project is obtained from the Internet. The coordinates then were constructed by using GML and simple features were created from the data. The maps created were in form of SVG files that contained point, line and polygon.

The process of graphical rendering took part after developing the GML document. The GML can be transform in three ways such as SVG, VML, and X3D. For this project, SVG is most suited since it transforms vector maps, which only involved simple features. The transformation was developed using SVG in XSLT format. The design is developed specific to the requirements and objective of the project.

Progress report was submitted during this phase. It described how the research is done, the facts and finding regarding the project, the result for the project progress and the discussion against the entire finding.

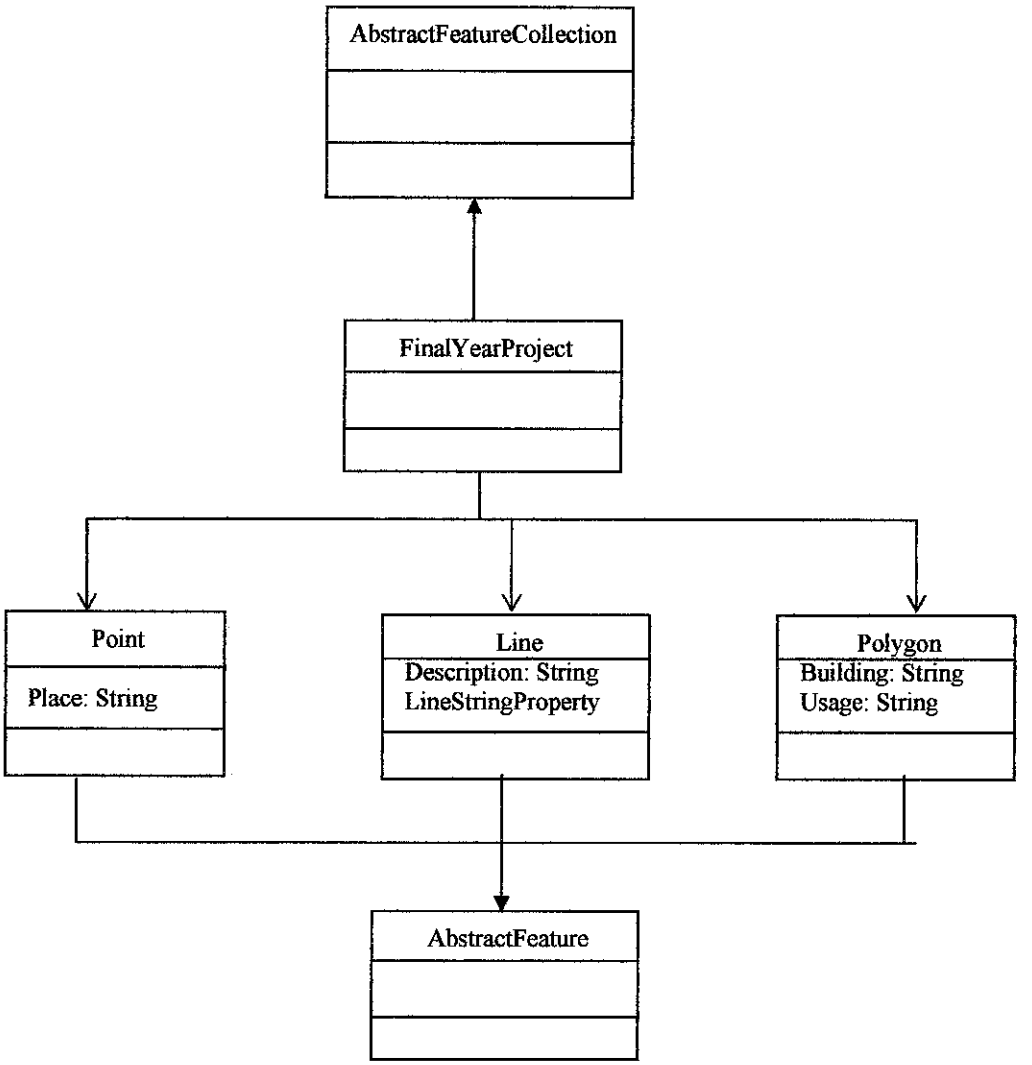


Figure 3.2 Unified Modeling Language (UML) Design

3.2.4 Implementation

Based on the XML standard, the GML handles both the geometry and the properties of the geographical elements; this allows the various data providers to share heterogeneous data sets and the users to access the data in a completely transparent way. From the application point of view, it simplifies and standardizes the operations in many sectors, from map building to data format transformation, from spatial query to geographical analysis, including the emerging applications in mobile systems. Since the GML data structure is XML-compliant, it can be transformed in a SVG (Scalable Vector Graphics) document format and then easily displayed on a standard web browser.

The implementation's main goal is the processing of GML documents and their visualization in a graphical way. A set of basic functionalities handling the graphical, the geographical and the thematic aspects has been developed for the project.

3.2.5 System Operation and Support

During system operation and support, the maintenance will maintain and enhance the system. Maintenance changes correct errors and adapt to changes in the environment. While enhancements provide new features and benefits. The objective during this phase is to maximize return on the IT/IS investment. The system developed shall be well-designed system that is reliable, maintainable, and scalable. During this stage, the drawback of the project will be identified and future enhancements will be made.

3.2 Tools And Hardware Required

There are many tools and hardware required for completing the project. These tools can significantly improve productive development and deployment of GML and increase the project performance. These are the best tools selected for the project:

- PC
- Arcview 3.2x/ Arcview 8.0
- Parser: Xerces-J v1.3.1
- Editor: XML Spy
- XSLT Processor: Saxon
- Graphics: Adobe SVG Viewer

CHAPTER 4

RESULT AND DISCUSSIONS

4. RESULT AND DISCUSSIONS

This chapter compiles the current findings of the project work. There are several important and informative facts and information which coming from journals and online resources. This chapter also discussed about the GML compatibility in developing map. The transformation from GML data to SVG also was discussed here in order to review the positive and negative impact in creating the map.

4.1 XML and GML in GIS

GML does not exactly define the rules of the attribute definition for non-geographical elements. These attributes can be expressed in the XML format, defining a specific application schema describing the structure and the types of the geographical and alphanumerical data used by the application. In this project, the focus is developing the GML document and transforms it into graphical representation by using XSLT and SVG.

4.1.1 Developing XSLT

XSL Transformations (XSLT) defines the process to transform the XML document into an Formatting Object (FO), and describes the semantics and rendering of each

formatting object and property. XSLT was designed to be more general than for use within XSL, allowing the transformation of documents into documents of any XML type, not only FO. In this project, the SVG was developed in together with XSLT in order to transform the GML document into the basic geometry features. Figure 4.1 shows the transformation of GML to SVG by using XSLT processor, which is Saxon.

Figure 4.1: Transformation of GML to SVG Using Saxon

4.1.2 Discussion on GML Map Result

There are several considerations to be taken care of, before developing on the map itself. Fields such as bounding box, coordinates, SVG size, and the map style need to be checked on, as wrong settings will make these resources unusable and wasted. This project basically focused on developing simple display of map (point, line or polygon) in a web browser using SVG.

There are three main geometry objects that have been used in this project which are point, line and polygon. The development of GML documents mainly used the GML version 1.0 since it already describes the simple way of the geometry features. Figure 4.2 shows the GML map produced in SVG. This map shows the basic geometry feature in GML, which is point. A Point is defined by a single coordinate.

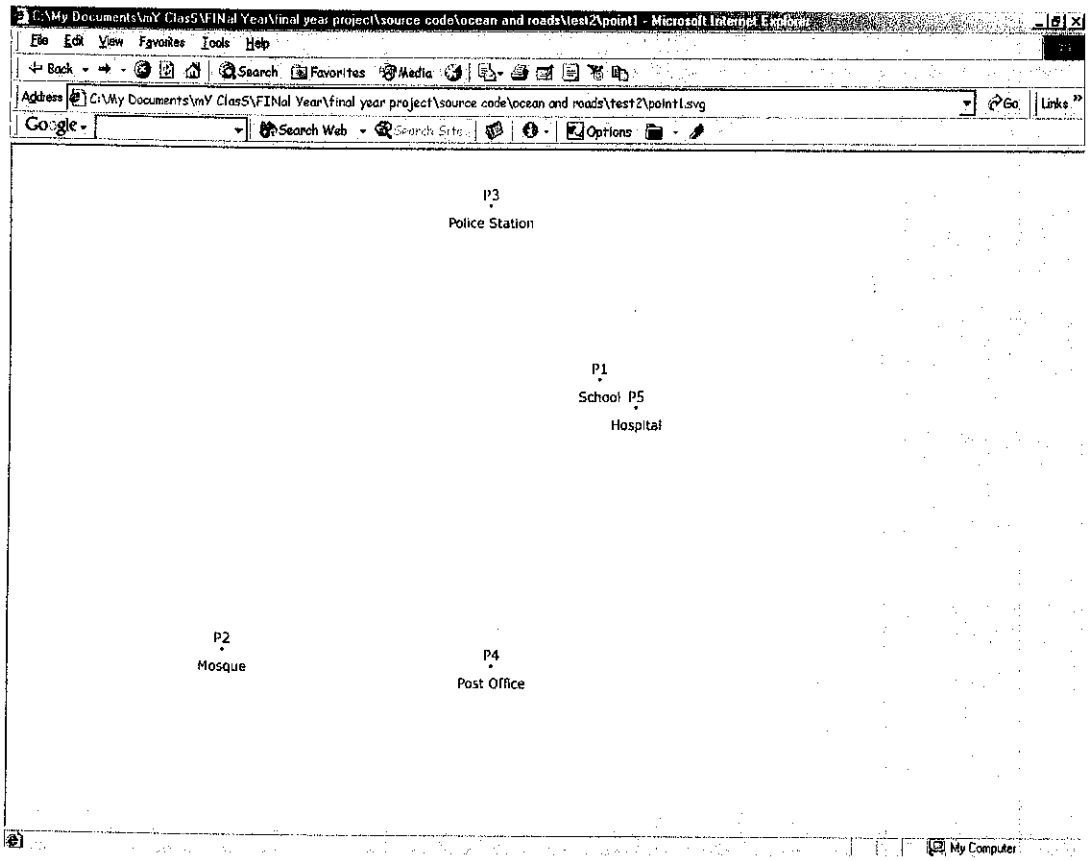


Figure 4.2: A simple display of locations in a web browser using SVG.

Referring to Figure 4.3, the map is produced from the coordinates that are plot up as line. Basically in GML there consist many type of line such as LineString and MultiLineString. LineString consist of two or more coordinates; with linear interpolation between them while MultiLineString consist of zero or many LineString. While in Figure 4.4 shows the map of a building that consists of polygon features.

4.1.3 Discussion on GML Compatibility

GML was developed with a number of explicit design goals, a few of which overlap the objectives of XML itself which provide a means of encoding spatial information for both data transport and data storage, especially in a wide-area Internet context. It also will be sufficiently extensible to support a wide variety of spatial tasks, from portrayal to analysis and establish the foundation for Internet GIS in an incremental and modular fashion.

GML allows for the efficient encoding of geo-spatial geometry (e.g. data compression) and provide easy-to-understand encodings of spatial information and spatial relationships, including those defined by the OGC Simple Features model. GML will be able to separate spatial and non-spatial content from data presentation (graphic or otherwise) besides permitting the easy integration of spatial and non-spatial data, especially for cases in which the nonspatial data is XML-encoded. GML is able to readily link spatial (geometric) elements to other spatial or non-spatial elements. It provides a set of common geographic modeling objects to enable interoperability of independently developed applications.

Features	GML	SVG
It is based on XML	√	√
It is text	√	√
It is stylable	√	√
It is concerned with the representation of the geographic data content	√	
It is a graphical vector data format		√
It encodes feature geometry and properties	√	

Table 4.1: Comparison between GML and SVG

4.2 GML Graphical Representation

Variety of graphical render programs is available for the various XML graphical formats, both as native in the web browser, as plug-in for the browser, as stand-alone viewer, or as library of functions. Geography Markup Language (GML) depends on a viewing format such as SVG for rendering GML objects to the browser. Graphical objects can be grouped, styled, transformed and composited into previously rendered objects.

4.2.1 Scalable Vector Graphic (SVG)

In the map styling process, the SVG Explorer application transforms the GML elements in SVG elements in order to represent them in a graphical way. This operation assigns to each GML tag, referred to the geometrical property of a geographical element, a SVG tag and establishes a sort of correspondence between these tags. This correspondence is not always unique but in some cases there are more than one SVG tag that can represent a GML geometrical property tag. Table 4.2 describes the SVG elements used by SVG Explorer in the map styling operation to produce the map. While Figure 4.5 shows the building map which has been zoom. This zooming capability is provided by SVG.

GML Geometry Element	SVG Element
Box	rec
Point	rec,circle or path
LineString	polyline, path
Polygon	path

Table 4.2: Correspondence between GML and SVG elements

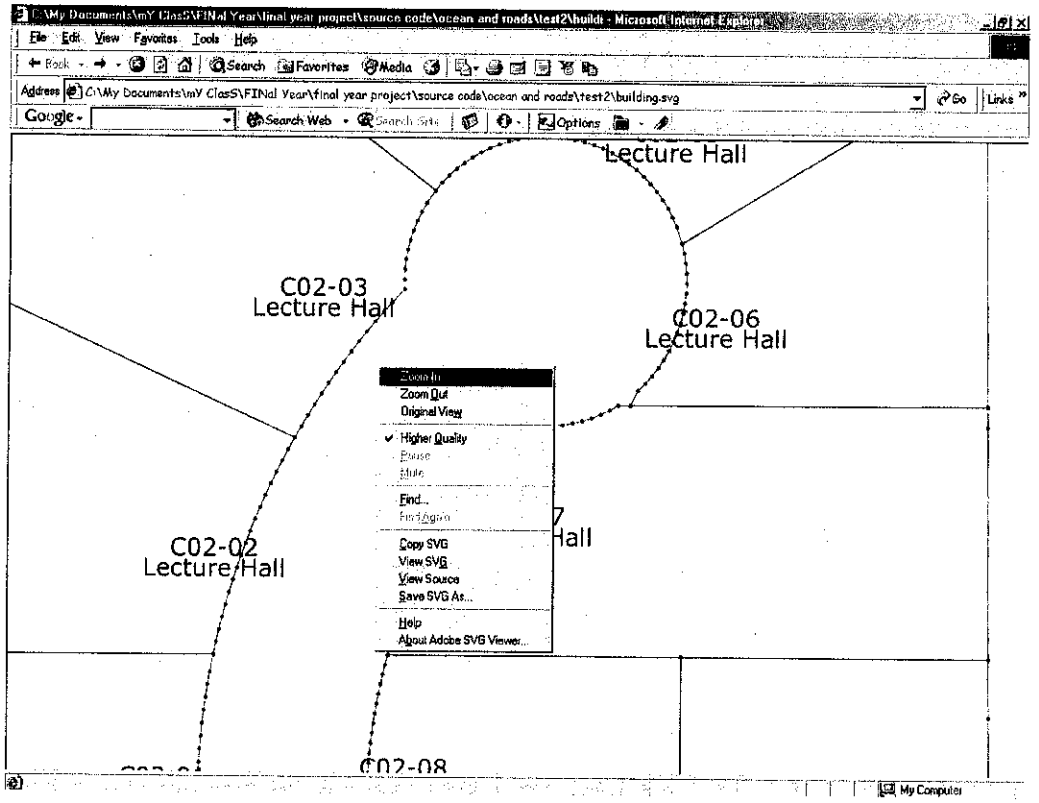


Figure 4.5: A simple display of Zoom In capability in a web browser using SVG.

4.2.2 Discussion on SVG Pro & Cons Result

SVG vector maps enhanced with customizable events come alive with dynamic capability not found on raster map publishing systems. In addition, interface designs are completely flexible, providing useful graphical Internet interfaces to legacy database systems. SVG-enabled browsers allow users to control their view with zoom and pan the same way they do so in GIS software. As the GIS community sees an expanding need for Internet extensions, SVG will play an important role in providing powerful, yet economical, solution.

SVG is mainly oriented to vector data but can define also raster data as JPEG, PNG files containing georeferenced images; with this approach the overlay of vector data on a raster data background is allowed. All the functionalities are operating in an ECMA Script environment, making use of libraries implementing base primitives working on GML and SVG data structures.

SVG also has its shortcomings. For example, although the Flash plug-in is prevalent among more than 95 percent of Internet users, a much lower percentage currently have the SVG plug-in [9]. These numbers will continue to grow, partly because Adobe is combining the SVG 3.0 plug-in with Adobe Acrobat. Overall, because SVG is an official W3C standard and based on XML, it's popularity is likely to compete with Flash in coming years.

SVG has been developed to handle vector graphics data and decreases some GIS facilities concerning specifically with the representation of geographical data. In particular it cannot handle the geographical coordinate system transformations and consequently cannot merge dynamically GML data with different coordinate systems on the same layer and on the same map. This transformation has to be done by specific functions operating on the GML data.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5. CONCLUSION AND RECOMMENDATIONS

The introduction of Geography Markup Language (GML) in Geography Information System has increased the geographic information distribution over the Internet. GML provides better quality maps are produced with editable features and query capabilities. GML will enable the geographic information to take part in the Spatial Web. GML also allows data integration easily between other data providers even if they are using different Geography Information System (GIS) package with its vendor specific storage format. GML is designed to support interoperability and does so through the provision of basic geometry tags (all systems that support GML use the same geometry tags), a common data model (features/properties), and a mechanism for creating and sharing application schemas. Most information communities will seek to enhance their interoperability by publishing their application schemas; interoperability may be further improved in some cases through the use of profiles.

This project relates on how XML and GML is use in Geographical Information System. Simple displays of map (point, line and polygon) have been developed and can be viewed in a web browser using SVG. Since the Internet revolution for all technology, this project really suits for current situation that needs more advanced technologies and ideas in geographic information areas. Too relying on old technology such as digitizing may give many disadvantages not only for user but also for the community. Specific applications can be created and

distributed to users for their own GIS needs. As the applications are running on the browser, no commercial GIS software must be purchased.

At present, the GML is oriented to vector data only and no specific element is defined to manage raster data (such as the geo-referenced images). This is a strong limitation because it does not allow the superimposition of vector data on a raster background. This restriction can be overcome specifying the raster image by means of an XML tag.

Several future expansions have been identified for this project. Since this project is more focused on the findings on how the spatial data is convert by using XML and GML, the next step can be developing the GML application that integrates the entire SVG maps. The features such as pan, and dynamic zooming also can be added to the system. The data also can be focused on UTP new academic building. Since the structure of the new building is complex and hard for GPS to connect with the satellite, the data can be obtained by using the theodolite technique. The new features of GML also can be considered such as complex features and voice representation.

In order to carry out this project, proper project management skills needed in order to ensure the project is developed according to requirement and within the timeframe. Proper sources need to be identified earlier since GML is a new technology in today's world. This is to avoid the lack of needed material and resources that will effect the development time. A combination of perseverance, endurance, persistence and carefulness is tremendously important in developing and completing this project.

Hopefully this project will benefit many organizations and overcome the drawback of digitizing map by using or manipulating the result of the research done in Geography Markup Language.

REFERENCES

[1] **"Geographic Information Systems"**, Feb 28 2003. <http://erg.usgs.gov/isb/pubs/gis_poster/>

[2] Prins, Mark. Feb 10, 2003 **"Is GML Only for Internet GIS?"** <http://www.directionsmag.com/article.php?article_id=280>

[3] OpenGIS Consortium Inc., 2003. **"Geography Markup Language (GML) Implementation Specification"**, OpenGIS Implementation Specification

[4] Ron Lake, 2000. **"Making Maps with Geography Markup Language (GML)"**, Galdos Systems, Inc.

[5] **"GIS Software"** Feb12, 2003. <http://www.gis.com/software/index.html >

[6] Vincent Dessard, 2002. **"GML & Web feature Server"**, IONIC Software s.a.

[7] Ehsan Muhammadi, Ali Aein, and Ali A. Alesheikh, 2003, "**Developing an Internet GIS Application Using GML"**, University of Technology Vali_asr St, Tehran, Iran

[8] George, Randy. Dec 2002, **"Maximize Online Mapping with SVG/XML"** < http://www.geoplace.com/gw/2002/0212/0212svg.asp>

[9] Fitzgerald, Brian. Jan 2003. **"Developed Online Demographic Visualization and Interaction With SVG"** <http://www.geoplace.com/gw/2003/0301/0301svg.asp>

[10] **"About Us - OGC",** Mar 13, 2003 <http://www.opengis.org/about/>

[11] **"OGC Approves GML 3, Demonstrates Web Services"** Mar 14, 2003 <http://www.geoplace.com/gw/2003/0304/0304nws.asp>

[12] Toon, Matt. Apr 99. **"XML As A New Web Language"** <http://www.geoplace.com/ma/1999/0499/499tch.asp>

[13] **"GML Technology"** Feb 4, 2003 <http://www.galdosinc.com/>

[14] **"OpenGIS® Geography Markup Language (GML) Implementation Specification, version 2.1.2, 2002"**, Open GIS Consortium, Inc.

[15] **"Location Organization Folder in GML",** OpenGIS© Consortium Discussion Paper 01-037, 2001, Open GIS Consortium, Inc.

[16] **"Project and Process Management - Methodology"** Mar 15, 2003 <http://www.sdmagazine.com/forums/thread.html?forumid=39&threadid=2343>

[17] Ron Lake, 2000. **"Location-Based Services & GML 2.0 - Laying the Geo-spatial Web Foundations"**, Galdos System, Inc.

[18] Ron Lake, 2000. **"GML 2.0 Enable the Geo-spatial Web"**, Galdos System, Inc.

[19] **"Geometry and Toplogy Schema"** , Jan 5, 2003 <http://www.ordnancesurvey.co.uk/oswebsite/xml/schema/OSGeometryTopology.xsd>

[20] Macgill, James. **"GML Example Map"**, Jan 6, 2003 <http://www.ccg.leeds.ac.uk/geotools/demos/gml/>

[21] http://webgisserver.cnuce.cnr.it/centroSI/progetti/pdf/mobile_gis_data_viewer.pdf

[22] Chimezie Ogbuji, June 04,2003. **"Visualizing XSLT in SVG"** <http://www.xml.com/pub/a/2003/06/04/xslt-svg.html>

[23] Mansfield ,Philip A. May 2002. **"Graphical Stylesheets, Using XSLT To Generate SVG "** <http://www.schemasoft.com/gcatools/gca2html/output/05-05-02.html >

[24] Anderson ,Geoff **"The Door Opens Open-Source GIS",** Feb 4, 2003 <http://www.geoplace.com/gw/2003/0306/0306opn1.asp>

[25] Dr. Winnie S. M. Tang, and Selwood, Jan Robert. **"The Development and Impact of Web-based Geographic Information Services"** Feb 4, 2003 <http://www.gisdevelopment.net/technology/gis/mi03002c.htm>

[26] J.D. Wilson, **"The Internet Dominates--Even in GIS",** Feb15, 2003 <http://www.geoplace.com/gw/2000/0200/0200gbw.asp >

[27] Berry, Joseph K. **"Use a Map-ematical Framework for GIS Modeling"** Feb15, 2003 <http://www.geoplace.com/gw/2004/0403/0403bmp.asp>

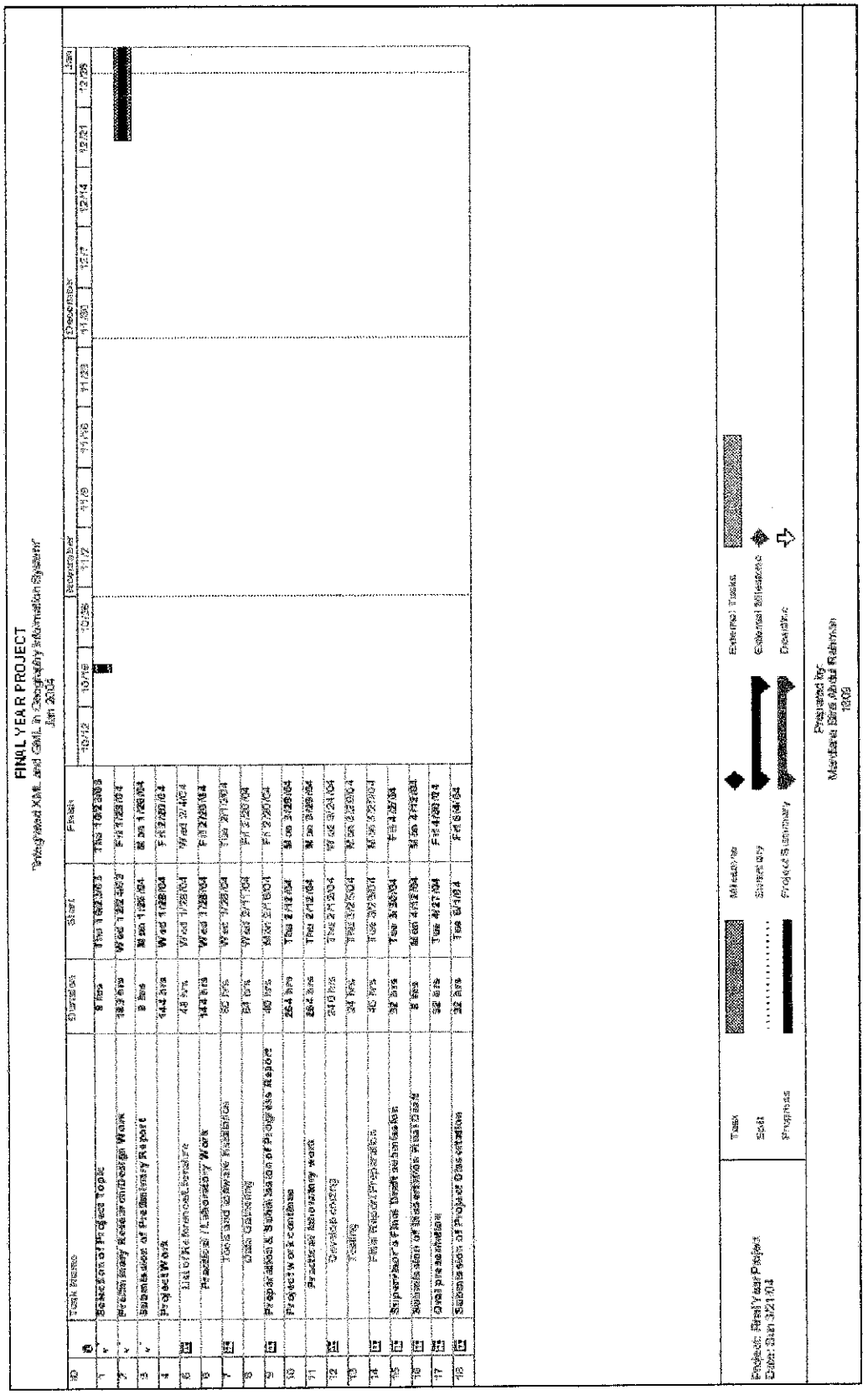
[28] Johnny, Marshall, **"Developing Internet Based GIS Applications",** INDUS Corporation, Vienna, VA <http://www.giscafe.com/technical_papers/Papers/paper058/>

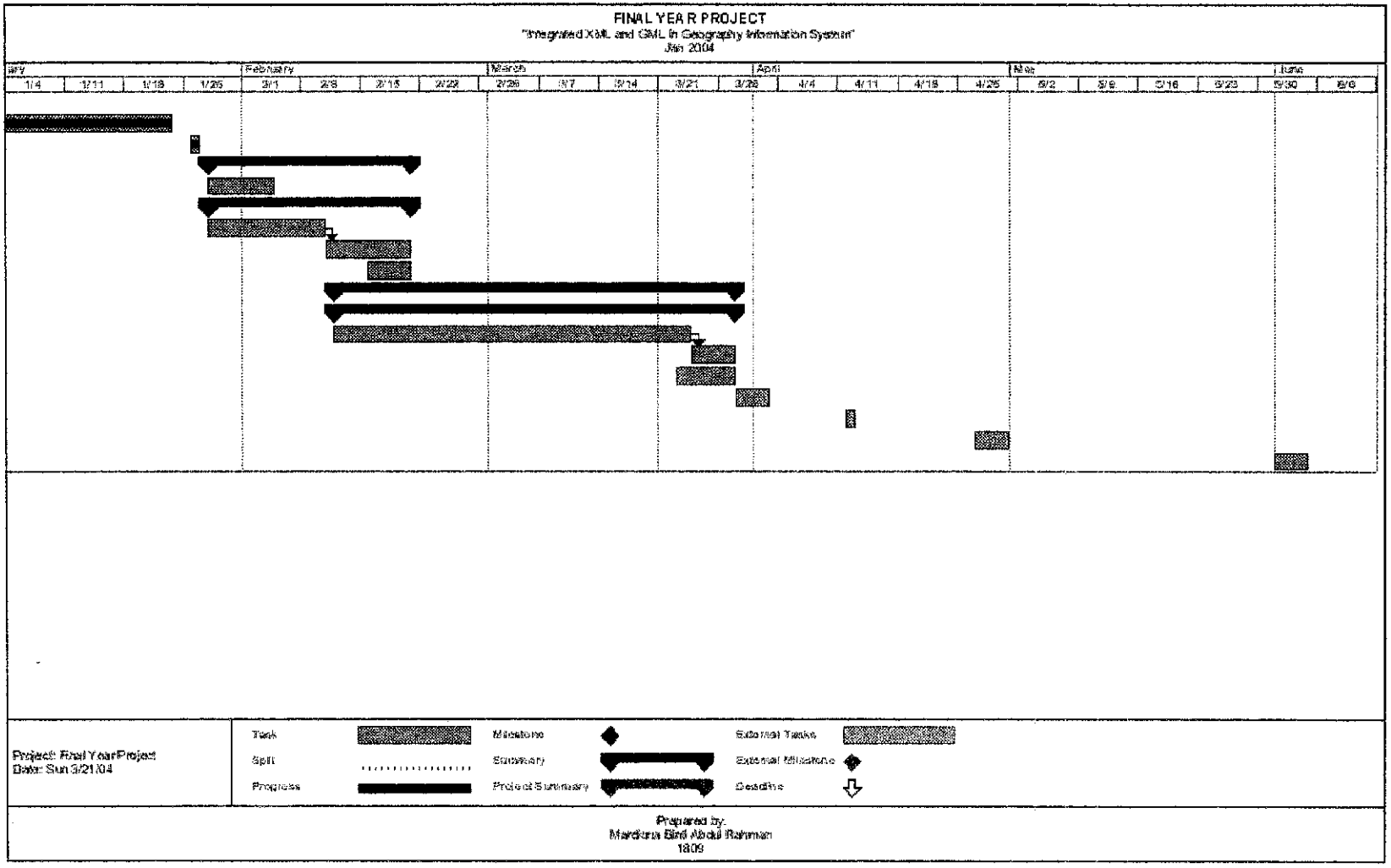
[29] Lewis D. Hopkins, Nikhil Kaza, and R. Varkki George Pallathucheril, July 2003, **"Planning Markup Language: Representing the Meanings of Plans and Regulations",** University of Illinois

[30] McKeown ,John and Grimson, Jane **"SVG: putting XML in the picture"**, Trinity College Dublin, Dublin, Ireland

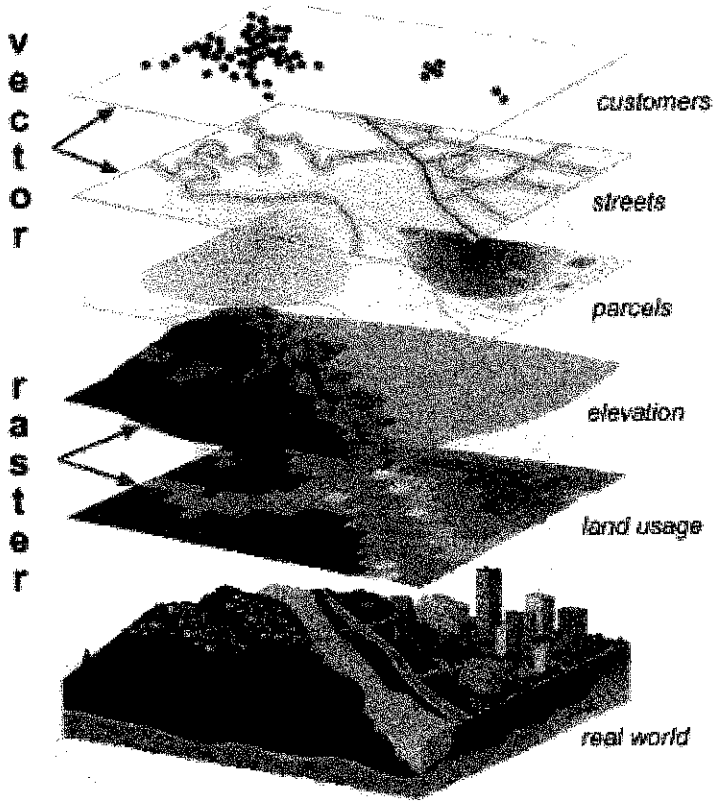
APPENDICES

APPENDIX 1 PROJECT SCHEDULE

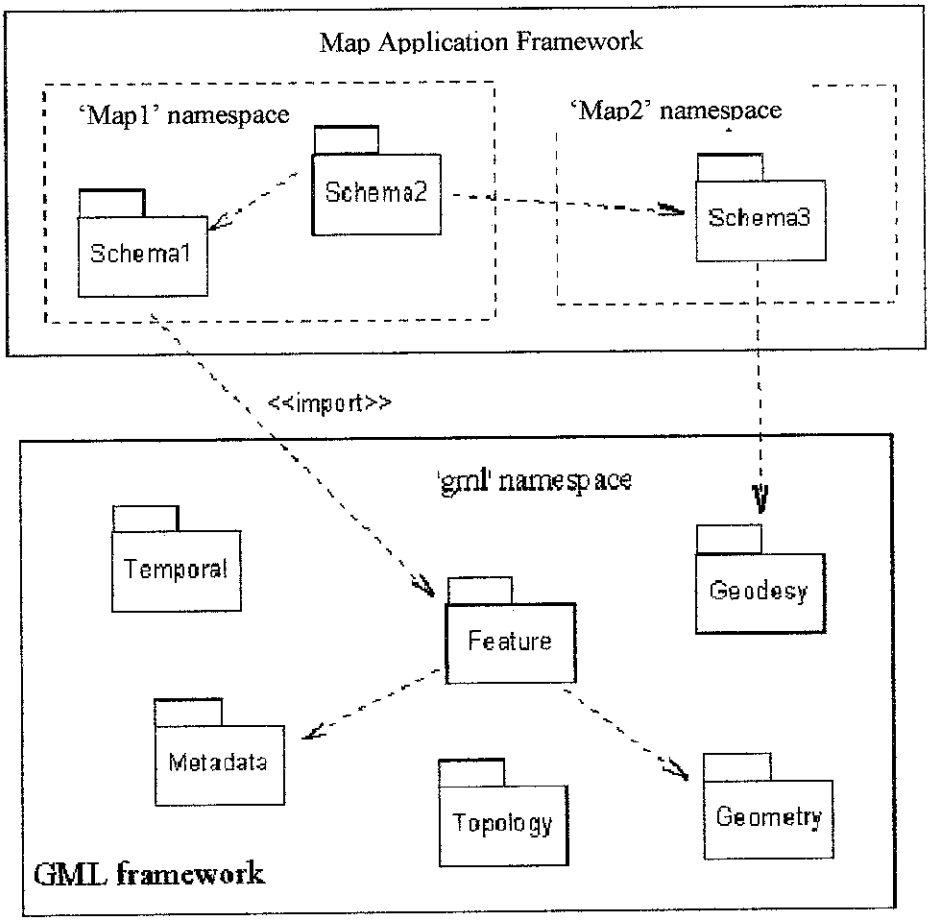



FINAL YEAR PROJECT
"Integrated XML and GML in Geography Information System"
Jan 2004
January
February
March
April
May
June
1/4
1/11
1/18
1/25
2/1
2/8
2/15
2/22
2/29
3/7
3/14
3/21
3/28
4/4
4/11
4/18
4/25
5/2
5/9
5/16
5/23
5/30
6/6
Project: Final Year Project
Date: Sun 5/2/04
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Prepared by:
Mardiana Binti Abdul Rahman
1809

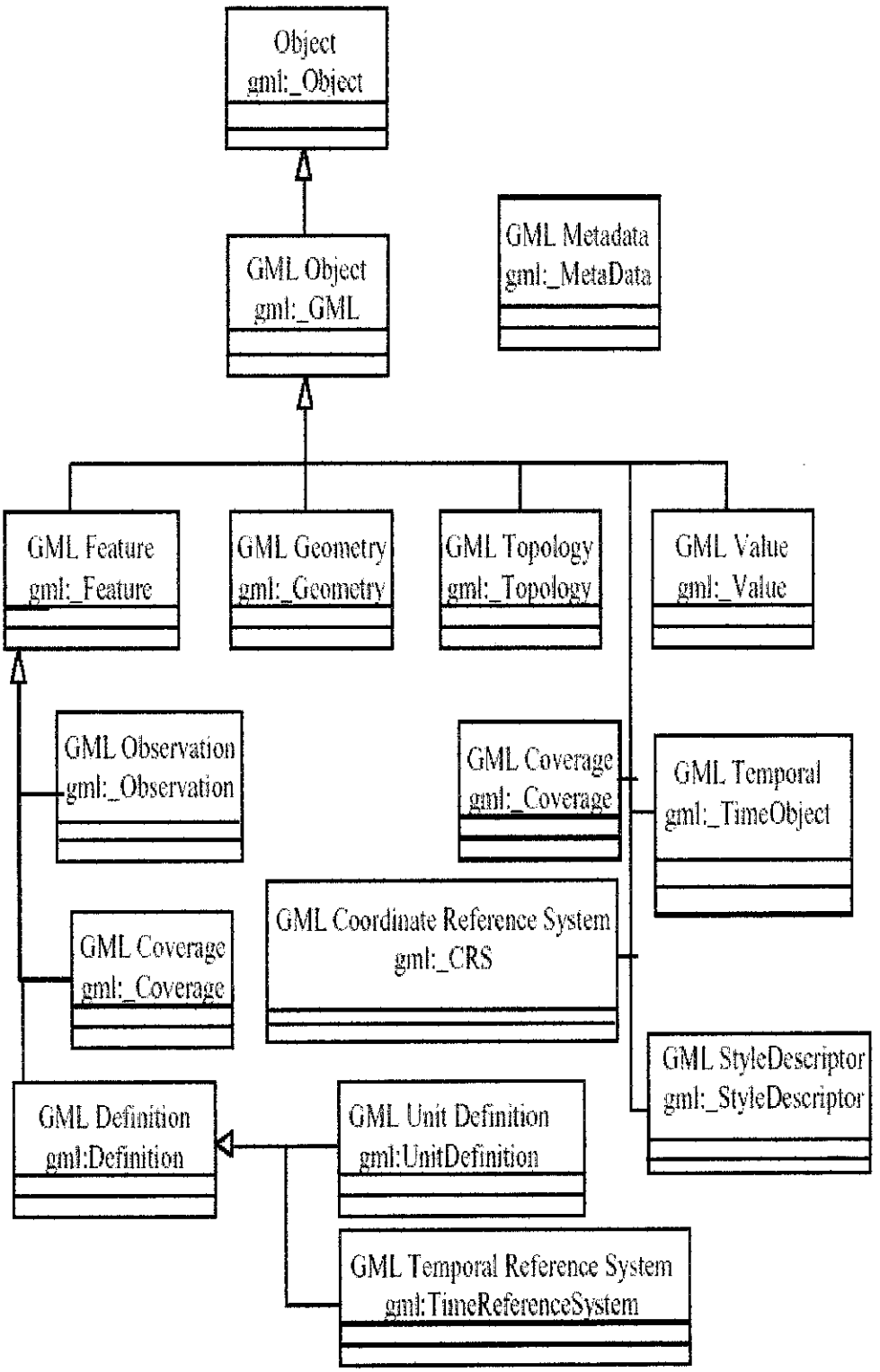
APPENDIX 2 GIS DATA STORAGE



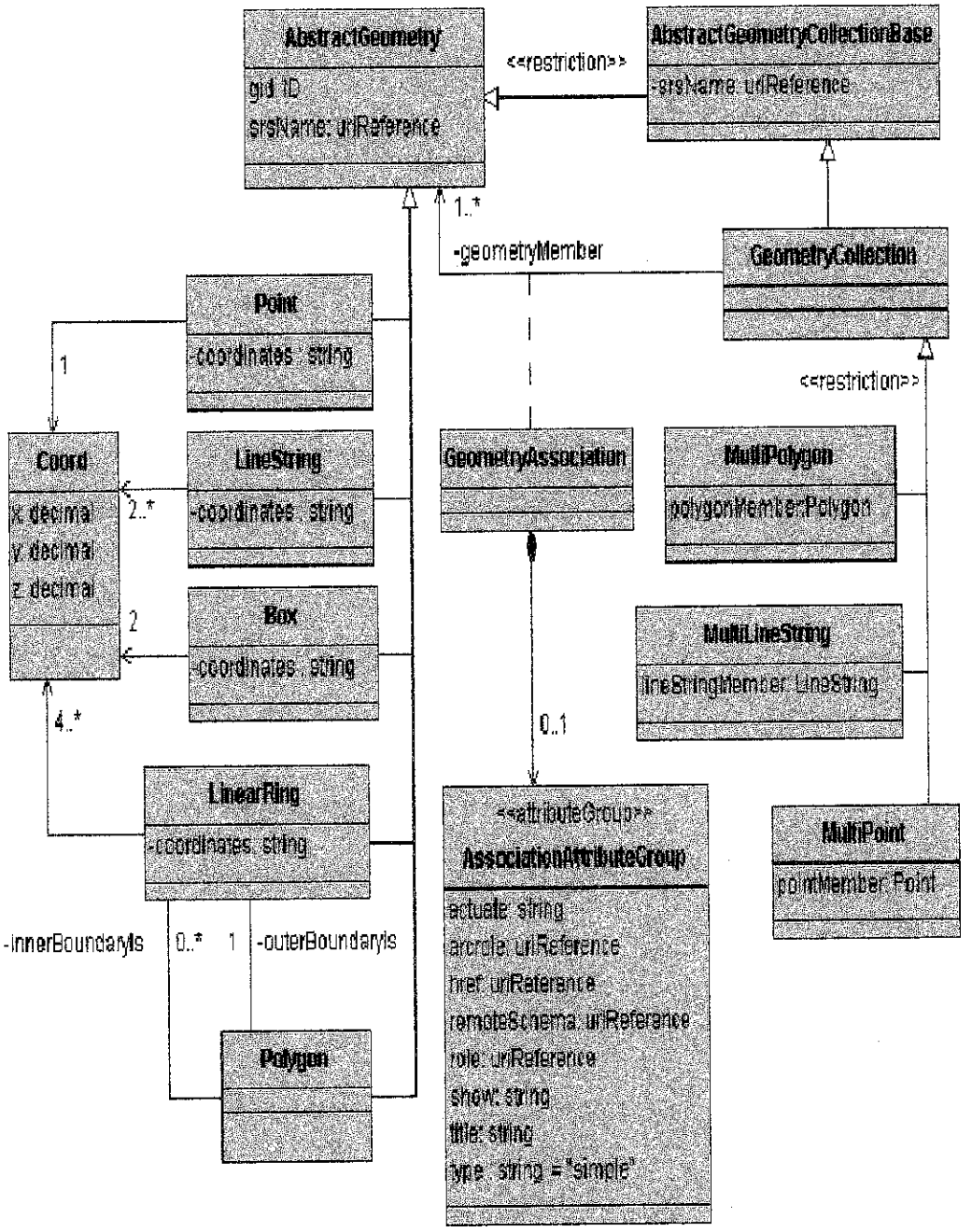
APPENDIX 3 GML AS A CORE FRAMEWORK



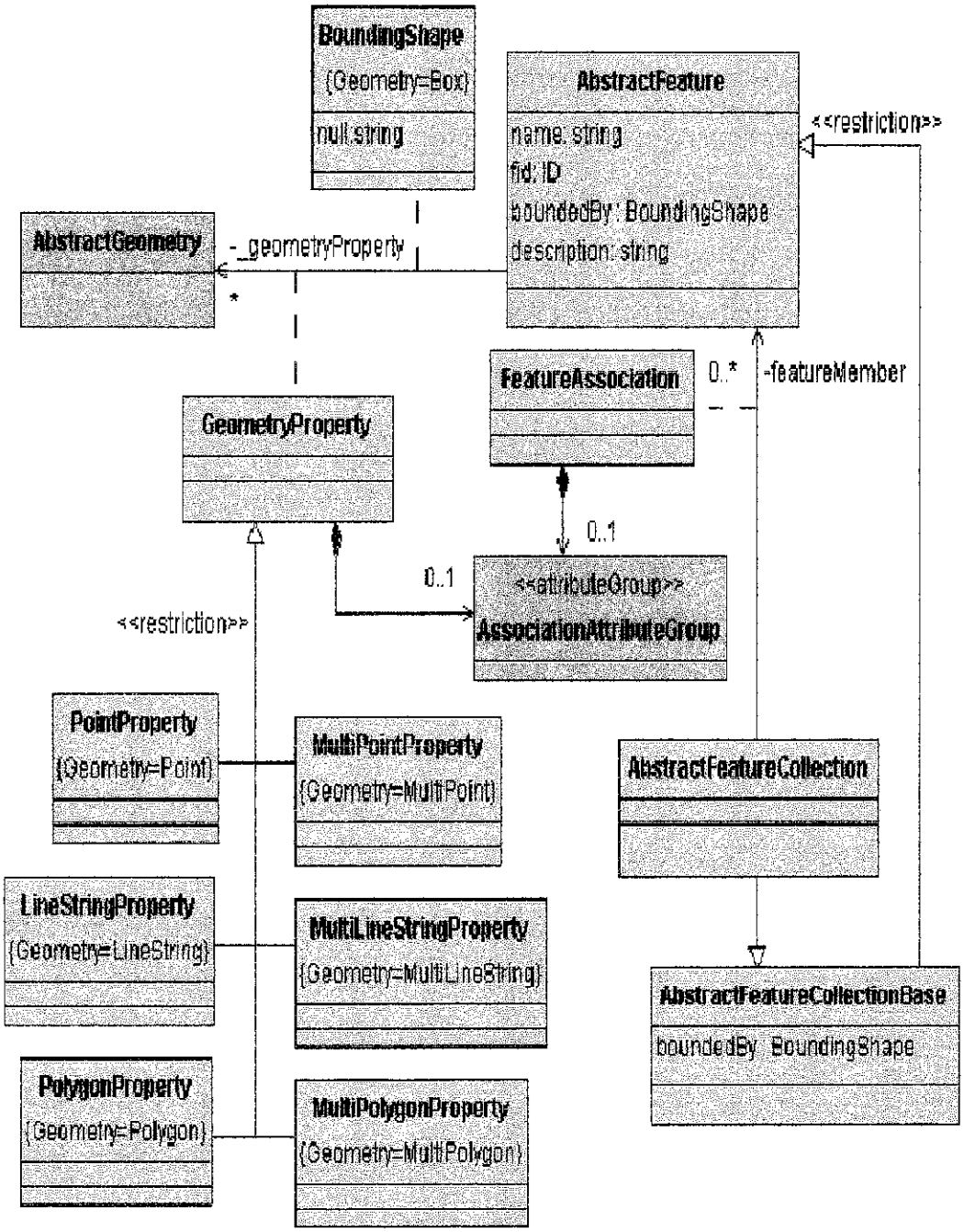
APPENDIX 4 GML CLASS HIERARCHY



APPENDIX 5 UML REPRESENTATION OF THE GEOMETRY SCHEMA



APPENDIX 6 UML REPRESENTATION OF THE FEATURE SCHEMA



APPENDIX 7 POINT.XML

```
<?xml version="1.0"?>
<!DOCTYPE FeatureCollection SYSTEM "gmlfeature.dtd">


<FeatureCollection typeName="Parcels">
    <boundedBy>
        <Box srsName="EPSG:XXXX">
            <coordinates>538336.0,1246940.0 530038.0,1247382.0</coordinates>
        </Box>
    </boundedBy>
    <featureMember typeName="Point11">
        <Feature typeName="Point1">
            <!--name>P1</name-->
            <property typeName="PLACE">School</property>
            <geometricProperty typeName="Boundary">
                <Point srsName="EPSG:XXXX">
                    <coordinates>
538623.899,1247250.484 538623.899,1247250.484    </coordinates>
                </Point>
            </geometricProperty>
        </Feature>
    </featureMember>
    <featureMember typeName="Point2">
        <Feature typeName="Point2">
            <!--name>P2</name-->
            <property typeName="PLACE">Mosque</property>
            <geometricProperty typeName="Boundary">
                <Point srsName="EPSG:XXXX">
                    <coordinates>
538372.295,1247075.120 538372.295,1247215.106 </coordinates>
                </Point>
            </geometricProperty>
        </Feature>
    </featureMember>
    <featureMember typeName="Point3">
        <Feature typeName="Point3">
            <!--name>P3</name-->
            <property typeName="PLACE">Police Station</property>
            <geometricProperty typeName="Boundary">
                <Point srsName="EPSG:XXXX">
                    <coordinates>
538550.272,1247361.096 538616.342,1247215.106 </coordinates>
                </Point>
            </geometricProperty>
        </Feature>
    </featureMember>
    <featureMember typeName="Point4">
        <Feature typeName="Point4">
            <!--name>P4</name-->
            <property typeName="PLACE">Post Office</property>
            <geometricProperty typeName="Boundary">
                <Point srsName="EPSG:XXXX">
                    <coordinates>
538552.299,1247065.532 538550.589,1247075.497    </coordinates>
                </Point>
            </geometricProperty>
        </Feature>
    </featureMember>
    <featureMember typeName="Point5">
        <Feature typeName="Point5">
            <!--name>P5</name-->
            <property typeName="PLACE">Hospital</property>
            <geometricProperty typeName="Boundary">
                <Point srsName="EPSG:XXXX">
                    <coordinates>
538648.414,1247232.950 538646.339,1247075.497    </coordinates>
                </Point>
            </geometricProperty>
        </Feature>
    </featureMember>
</FeatureCollection>
```

APPENDIX 8 LINE.XML

```
<?xml version="1.0" encoding="utf-8"?>
<!--        Project                 : Integrated XML and GML in Geography Information System
            Developer               : Mardiana Binti Abdul Rahman
            ID                      : 1809
            Institution             : University Technologi PETRONAS
            Document Description    : This document consist of GML document for describing line
                                      feature-->

<!-- SCHEMA DECLARATION -->

<exp:FeatureCollection xmlns="http://www.opengis.net/exp" xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance" xmlns:exp="http://www.opengis.net/examples"
xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.opengis.net/examples fyp.xsd">

<!--DEFINING BOUNDINGBOX -->

    <gml:boundedBy xmlns:gml="http://www.opengis.net/gml">
        <gml:Box>
            <gml:coordinates>482000,5449000 505000,5472000</gml:coordinates>
        </gml:Box>
    </gml:boundedBy>

<!-- BODY -->

    <gml:featureMember>
        <exp:RL1U fid="RL1U10644">
            <gml:description>Roads rL1U</gml:description>
            <gml:name/>
            <gml:centerLineOf xmlns:gml="http://www.opengis.net/gml">
                <gml:LineString>
                    <gml:coordinates>492298.999999489,5453174.99963225
492314.999999489,5453173.99963225 492334.99999949,5453172.99963225
492352.999999492,5453172.99963225 492373.999999495,5453171.99963225
492395.999999489,5453170.99963225 492415.999999499,5453168.99963225
492433.9999995,5453168.99963225 492452.999999493,5453167.99963225
492454.999999499,5453167.99963224</gml:coordinates>
                </gml:LineString>
            </gml:centerLineOf>
        </exp:RL1U>
    </gml:featureMember>
    <gml:featureMember>
        <exp:RL1U fid="RL1U10645">
            <gml:description>Roads rL1U</gml:description>
            <gml:name/>
            <gml:centerLineOf xmlns:gml="http://www.opengis.net/gml">
                <gml:LineString>
                    <gml:coordinates>492454.999999499,5453167.99963224
492472.999999494,5453167.99963224 492493.999999503,5453166.99963225
492514.999999501,5453166.99963225 492535.999999503,5453166.99963225
492554.999999507,5453165.99963225 492563.999999508,5453165.99963225</gml:coordinates>
                </gml:LineString>
            </gml:centerLineOf>
        </exp:RL1U>
    </gml:featureMember>
    <gml:featureMember>
        <exp:RL1U fid="RL1U10647">
            <gml:description>Roads rL1U</gml:description>
            <gml:name/>
            <gml:centerLineOf xmlns:gml="http://www.opengis.net/gml">
                <gml:LineString>
                    <gml:coordinates>492565.999999504,5453265.99963228
492546.999999504,5453265.99963227 492526.999999505,5453265.99963227
492505.999999504,5453266.99963227 492484.999999497,5453266.99963228
492465.999999498,5453267.99963227 492452.9999995,5453267.99963228</gml:coordinates>
                </gml:LineString>
            </gml:centerLineOf>
        </exp:RL1U>
    </gml:featureMember>
    <gml:featureMember>
        <exp:RL1U fid="RL1U10648">
            <gml:description>Roads rL1U</gml:description>
            <gml:name/>
```

<gml:centerLineOf xmlns:gml="http://www.opengis.net/gml">
<gml:LineString>
<gml:coordinates>492302.99999949,5453325.99963229
492296.99999949,5453325.99963229 492276.999999485,5453325.99963229
492256.999999478,5453326.99963229 492234.999999479,5453327.99963229
492212.999999485,5453328.99963229 492191.999999482,5453329.99963229
492171.999999476,5453330.99963229 492149.99999948,5453331.99963229
492127.999999481,5453332.99963229 492125.999999471,5453332.99963229</gml:coordinates>
</gml:LineString>
</gml:centerLineOf>
</exp:RL1U>
</gml:featureMember>
<gml:featureMember>
<exp:RL1U fid="RL1U10655">
<gml:description>Roads rL1U</gml:description>
<gml:name/>
<gml:centerLineOf xmlns:gml="http://www.opengis.net/gml">
<gml:LineString>
<gml:coordinates>491598.99999944,5453162.99963224
491600.999999436,5453162.99963224 491620.999999445,5453160.99963224
491641.999999438,5453158.99963224 491662.999999444,5453157.99963225
491682.999999449,5453156.99963224 491702.999999451,5453154.99963224
491721.999999451,5453152.99963224 491742.999999449,5453150.99963224
491760.999999451,5453149.99963224</gml:coordinates>
</gml:LineString>
</gml:centerLineOf>
</exp:RL1U>
</gml:featureMember>
<gml:featureMember>
<exp:RL1U fid="RL1U10736">
<gml:description>Roads rL1U</gml:description>
<gml:name/>
<gml:centerLineOf xmlns:gml="http://www.opengis.net/gml">
<gml:LineString>
<gml:coordinates>492651.999999508,5452546.99963208
492651.999999513,5452531.99963207 492650.999999508,5452510.99963207
492650.999999512,5452495.99963206</gml:coordinates>
</gml:LineString>
</gml:centerLineOf>
</exp:RL1U>
</gml:featureMember>
<gml:featureMember>
<exp:RL1U fid="RL1U10739">
<gml:description>Roads rL1U</gml:description>
<gml:name/>
<gml:centerLineOf xmlns:gml="http://www.opengis.net/gml">
<gml:LineString>
<gml:coordinates>492648.999999511,5452386.99963203
492648.99999951,5452368.99963203 492647.999999512,5452348.99963202
492647.99999951,5452338.99963202</gml:coordinates>
</gml:LineString>
</gml:centerLineOf>
</exp:RL1U>
</gml:featureMember>
</exp:FeatureCollection>

APPENDIX 9 POLYGON.XML

```
<?xml version="1.0"?>
<!DOCTYPE FeatureCollection SYSTEM "gmlfeature.dtd">

<!--        Project                 : Integrated XML and GML in Geography Information System
            Developer               : Mardiana Binti Abdul Rahman
            ID                      : 1809
            Institution             : University Technologi PETRONAS
            Document Description    : This document consist of GML document for describing polygon-->

<FeatureCollection typeName="Parcels">
    <boundedBy>
        <Box srsName="EPSG:XXXX">
            <coordinates>538336.0,1246940.0 530038.0,1247382.0</coordinates>
        </Box>
    </boundedBy>
    <featureMember typeName="Parcel1">
        <Feature typeName="Parcel1">
            <name>93-21-21-155-027</name>
            <property typeName="BUILDING">C02-05</property>
            <property typeName="USAGE">Lecture Hall</property>
            <geometricProperty typeName="Boundary">
                <Polygon srsName="EPSG:XXXX">
                    <outerBoundaryIs>
                        <LinearRing>
                            <coordinates>
538668.269,1247362.009 538782.855,1247362.896 538762.840,1247282.225 538655.423,1247217.867
538653.402,1247223.764 538651.955,1247226.939 538650.290,1247230.007 538648.414,1247232.950
538646.339,1247235.756 538644.072,1247238.409 538641.626,1247240.899 538639.013,1247243.211
538636.244,1247245.336 538633.334,1247247.262 538630.296,1247248.981 538627.146,1247250.484
538623.899,1247251.763 538620.571,1247252.813 538617.178,1247253.627 538616.342,1247361.607
538668.269,1247362.009    </coordinates>
                        </LinearRing>
                    </outerBoundaryIs>
                </Polygon>
            </geometricProperty>
        </Feature>
    </featureMember>
    <featureMember typeName="Parcel2">
        <Feature typeName="Parcel2">
            <name>93-21-21-155-026</name>
            <property typeName="BUILDING">C02-04</property>
            <property typeName="USAGE">Lecture Hall</property>
            <geometricProperty typeName="Boundary">
                <Polygon srsName="EPSG:XXXX">
                    <outerBoundaryIs>
                        <LinearRing>
                            <coordinates>
538550.272,1247361.096 538616.342,1247361.607 538617.178,1247253.627 538611.003,1247254.487
538607.516,1247254.631 538604.028,1247254.531 538600.555,1247254.187 538597.114,1247253.602
538593.723,1247252.779 538590.397,1247251.721 538587.153,1247250.434 538584.007,1247248.924
538580.974,1247247.197 538578.069,1247245.264 538575.305,1247243.132 538572.697,1247240.813
538570.257,1247238.318 538567.998,1247235.658 538410.315,1247355.013 538550.311,1247356.096
538550.272,1247361.096    </coordinates>
                        </LinearRing>
                    </outerBoundaryIs>
                </Polygon>
            </geometricProperty>
        </Feature>
    </featureMember>
    <featureMember typeName="Parcel3">
        <Feature typeName="Parcel3">
            <name>93-21-21-155-025</name>
            <property typeName="BUILDING">C02-03</property>
            <property typeName="USAGE">Lecture Hall</property>
            <geometricProperty typeName="Boundary">
                <Polygon srsName="EPSG:XXXX">
                    <outerBoundaryIs>
                        <LinearRing>
                            <coordinates>
538410.315,1247355.013 538567.998,1247235.658 538564.910,1247231.296 538563.153,1247228.281
538561.611,1247225.150 538560.291,1247221.919 538559.199,1247218.605 538558.341,1247215.222
538557.722,1247211.787 538557.343,1247208.318 538557.207,1247204.831 538557.315,1247201.342
```

```
538552.932,1247196.717 538549.940,1247193.423 538547.005,1247190.077 538544.130,1247186.680
538541.314,1247183.233 538538.559,1247179.738 538535.865,1247176.195 538533.234,1247172.606
538530.665,1247168.971 538528.161,1247165.293 538525.721,1247161.571 538523.346,1247157.806
538521.037,1247154.002 538518.796,1247150.157 538372.295,1247215.106 538370.316,1247354.703
538410.315,1247355.013
                        </coordinates>
                    </LinearRing>
                </outerBoundaryIs>
            </Polygon>
        </geometricProperty>
    </Feature>
</featureMember>
<featureMember typeName="Parcel4">
    <Feature typeName="Parcel4">
        <name>93-21-21-155-028</name>
        <property typeName="BUILDING">C02-06</property>
        <property typeName="USAGE">Lecture Hall</property>
        <geometricProperty typeName="Boundary">
            <Polygon srsName="EPSG:XXXX">
                <outerBoundaryIs>
                    <LinearRing>
                        <coordinates>
538762.840,1247282.225 538763.099,1247255.226 538763.992,1247162.230 538637.547,1247161.948
538640.239,1247167.097 538642.777,1247169.492 538645.142,1247172.059 538647.321,1247174.785
538649.306,1247177.656 538651.085,1247180.658 538652.650,1247183.777 538653.994,1247186.998
538655.110,1247190.305 538655.993,1247193.681 538656.638,1247197.111 538657.042,1247200.578
538657.204,1247204.064 538657.121,1247207.553 538656.796,1247211.027 538656.229,1247214.471
538655.423,1247217.867 538762.840,1247282.225
                        </coordinates>
                    </LinearRing>
                </outerBoundaryIs>
            </Polygon>
        </geometricProperty>
    </Feature>
</featureMember>
<featureMember typeName="Parcel5">
    <Feature typeName="Parcel5">
        <name>93-21-21-155-024</name>
        <property typeName="BUILDING">C02-02</property>
        <property typeName="USAGE">Lecture Hall</property>
        <geometricProperty typeName="Boundary">
            <Polygon srsName="EPSG:XXXX">
                <outerBoundaryIs>
                    <LinearRing>
                        <coordinates>
538372.295,1247215.106 538518.796,1247150.157 538516.515,1247146.080 538514.412,1247142.157
538512.378,1247138.199 538510.413,1247134.206 538508.518,1247130.179 538506.694,1247126.119
538504.941,1247122.028 538503.259,1247117.908 538501.650,1247113.758 538500.114,1247109.581
538498.650,1247105.378 538497.260,1247101.150 538495.944,1247096.899 538494.703,1247092.625
538493.536,1247088.330 538492.445,1247084.016 538491.428,1247079.683 538490.488,1247075.333
538374.279,1247075.120 538372.295,1247215.106
                        </coordinates>
                    </LinearRing>
                </outerBoundaryIs>
            </Polygon>
        </geometricProperty>
    </Feature>
</featureMember>
<featureMember typeName="Parcel6">
    <Feature typeName="Parcel6">
        <name>93-21-21-155-029</name>
        <property typeName="BUILDING">C02-07</property>
        <property typeName="USAGE">Lecture Hall</property>
        <geometricProperty typeName="Boundary">
            <Polygon srsName="EPSG:XXXX">
                <outerBoundaryIs>
                    <LinearRing>
                        <coordinates>
538763.992,1247162.230 538764.059,1247155.230 538764.827,1247075.234 538656.132,1247075.368
538552.299,1247075.497 538553.287,1247078.943 538554.288,1247082.217 538555.347,1247085.472
538556.462,1247088.708 538557.634,1247091.925 538558.861,1247095.121 538560.144,1247098.294
538561.483,1247101.445 538562.876,1247104.572 538564.323,1247107.674 538565.824,1247110.750
538567.379,1247113.800 538568.987,1247116.822 538570.647,1247119.816 538572.360,1247122.780
538574.123,1247125.714 538575.938,1247128.617 538577.803,1247131.487 538579.718,1247134.325
538581.682,1247137.128 538583.695,1247139.897 538585.756,1247142.631 538587.864,1247145.328
```

```
538590.019,1247147.988 538592.220,1247150.609 538594.466,1247153.192 538596.757,1247155.736
538603.167,1247154.795 538606.653,1247154.635 538610.142,1247154.718 538613.617,1247155.044
538617.060,1247155.612 538620.456,1247156.419 538623.787,1247157.461 538627.037,1247158.732
538630.190,1247160.227 538633.232,1247161.939 538637.547,1247161.948 538763.992,1247162.230
                                        </coordinates>
                        </LinearRing>
                    </outerBoundaryIs>
                </Polygon>
            </geometricProperty>
        </Feature>
    </featureMember>
<featureMember typeName="Parcel7">
        <Feature typeName="Parcel7">
            <name>93-21-21-155-030</name>
            <property typeName="BUILDING">C02-08</property>
            <property typeName="USAGE">Lecture Hall</property>
            <geometricProperty typeName="Boundary">
                <Polygon srsName="EPSG:XXXX">
                    <outerBoundaryIs>
                        <LinearRing>
                            <coordinates>
538552.299,1247075.497 538656.132,1247075.368 538655.983,1246955.209 538545.983,1246955.179
538545.207,1247021.192 538545.204,1247024.789 538545.261,1247028.211 538545.379,1247031.632
538545.556,1247035.051 538545.793,1247038.466 538546.089,1247041.876 538546.445,1247045.281
538546.860,1247048.679 538547.334,1247052.069 538547.868,1247055.451 538548.460,1247058.822
538549.111,1247062.183 538549.821,1247065.532 538550.589,1247068.868 538551.415,1247072.190
538552.299,1247075.497     </coordinates>
                        </LinearRing>
                    </outerBoundaryIs>
                </Polygon>
            </geometricProperty>
        </Feature>
    </featureMember>
    <featureMember typeName="Parcel8">
        <Feature typeName="Parcel8">
            <name>93-21-21-155-031</name>
            <property typeName="BUILDING">C02-09</property>
            <property typeName="USAGE">Lecture Hall</property>
            <geometricProperty typeName="Boundary">
                <Polygon srsName="EPSG:XXXX">
                    <outerBoundaryIs>
                        <LinearRing>
                            <coordinates>
538656.132,1247075.368 538764.827,1247075.234 538765.019,1247055.235 538765.978,1246955.240
538655.983,1246955.209 538656.132,1247075.368    </coordinates>
                        </LinearRing>
                    </outerBoundaryIs>
                </Polygon>
            </geometricProperty>
        </Feature>
    </featureMember>
    <featureMember typeName="Parcel9">
        <Feature typeName="Parcel9">
            <name>93-21-21-155-023</name>
            <property typeName="BUILDING">C02-01</property>
            <property typeName="USAGE">Lecture Hall</property>
            <geometricProperty typeName="Boundary">
                <Polygon srsName="EPSG:XXXX">
                    <outerBoundaryIs>
                        <LinearRing>
                            <coordinates>
538374.279,1247075.120 538490.488,1247075.333 538489.323,1247069.343 538488.563,1247064.958
538487.880,1247060.560 538487.274,1247056.151 538486.745,1247051.732 538486.293,1247047.305
538485.919,1247042.870 538485.622,1247038.429 538485.402,1247033.984 538485.260,1247029.536
538485.196,1247025.086 538485.210,1247020.635 538485.979,1246955.162 538375.979,1246955.132
538374.279,1247075.120
                            </coordinates>
                        </LinearRing>
                    </outerBoundaryIs>
                </Polygon>
            </geometricProperty>
        </Feature>
    </featureMember>
</FeatureCollection>
```

APPENDIX 10 BUILDING.XSL

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<!--    Project                 : Integrated XML and GML in Geography Information System
        Developer               : Mardiana Binti Abdul Rahman
        ID                      : 1809
        Institution             : University Technologi PETRONAS
        Document Description    : This document consist of eXtensible Stylesheet Language
                                  Transformation(XSLT). This stylesheet will transform the GML
                                  into SVG and will be viewed in web browser -->

    <xsl:preserve-space elements="*"/>
    <!-- The root template -->
    <xsl:template match="/">
        <!-- Use xsl:element to add the computed attribute value for the viewBox -->
        <xsl:element name="svg" namespace="">
            <xsl:attribute name="width">800</xsl:attribute>
            <xsl:attribute name="height">600</xsl:attribute>
            <xsl:attribute name="viewBox"><xsl:call-template name="get-bounding-box"/></xsl:attribute>
            <!-- The feature template adds the text -->
            <xsl:apply-templates select="//Feature"/>
            <!-- This template does the drawing of the parcels -->
            <xsl:apply-templates select="//Polygon/outerBoundaryIs/LinearRing"/>
            <!-- Finally we mark the points -->
            <xsl:apply-templates select="//Feature//coordinates"/>
        </xsl:element>
    </xsl:template>
    <!--
=============================================================================
   transform-x: Transforms the x (right value - mathematical coordinate system)
   Parameters: coordinate pair     comma separated list of coordinates x,y
           x-offset        x-coordinate (right-value) of the upper left corner
           scale           scale; assumes a screen resolution of 72 dpi
=============================================================================-->
    <xsl:template name="transform-x">
        <xsl:param name="coordinate-pair"/>
        <xsl:param name="x-offset" select="538336"/>
        <xsl:param name="scale" select="2000"/>
        <xsl:value-of select="(substring-before($coordinate-pair,',') - $x-offset) * 2835 div $scale"/>
    </xsl:template>
    <!--
=============================================================================
   transform-y: Transforms the y (up value - mathematical coordinate system)
   Parameters: coordinate pair     comma separated list of coordinates x,y
           y-offset        y-coordinate (up-value) of the upper left corner
           scale           scale; assumes a screen resolution of 72 dpi
=============================================================================-->
    <xsl:template name="transform-y">
        <xsl:param name="coordinate-pair"/>
        <xsl:param name="y-offset" select="1246940"/>
        <xsl:param name="scale" select="2000"/>
        <xsl:value-of select="(substring-after($coordinate-pair,',') - $y-offset) * -2835 div $scale + 600"/>
    </xsl:template>
<!--=========================================================================
   get-bounding-box: returns space separated list of the (screen!) coordinates of the
   upper left and bottom right corner.
=========================================================================-->
    <xsl:template name="get-bounding-box">
        <!-- First we concatenate all coordinates into one list -->
        <xsl:variable name="all-coords">
            <xsl:for-each select="//Feature//coordinates">
                <xsl:value-of select="."/>
            </xsl:for-each>
        </xsl:variable>
        <!-- Now we get the minimum x, y etc. one after the other -->
        <xsl:call-template name="get-min-x">
            <xsl:with-param name="list" select="$all-coords"/>
            <xsl:with-param name="cval" select="100000"/>
        </xsl:call-template>
        <!-- And add whitespace as separator -->
        <xsl:text/>
```

```
        <xsl:call-template name="get-min-y">
            <xsl:with-param name="list" select="$all-coords"/>
            <xsl:with-param name="cval" select="100000"/>
        </xsl:call-template>
        <xsl:text/>
        <xsl:call-template name="get-max-x">
            <xsl:with-param name="list" select="$all-coords"/>
            <xsl:with-param name="cval" select="0"/>
        </xsl:call-template>
        <xsl:text/>
        <xsl:call-template name="get-max-y">
            <xsl:with-param name="list" select="$all-coords"/>
            <xsl:with-param name="cval" select="0"/>
        </xsl:call-template>
    </xsl:template>
<!=============================================================================
  get-min-x: gets the minimum x (screen-)coordinate of a list of coordinates.
  Parameters:  list      space-separated list of coordinate pairs x,y.
               cval      used for recursion, must be initialized to a big value
==============================================================================>
    <xsl:template name="get-min-x">
        <xsl:param name="list"/>
        <xsl:param name="cval"/>
        <xsl:variable name="wlist" select="normalize-space($list)"/>
        <xsl:variable name="cpair" select="substring-before($wlist,' ')"/>
        <!-- First we transform from real world to screen coordinates and store x-value -->
        <xsl:variable name="x">
            <xsl:call-template name="transform-x">
                <xsl:with-param name="coordinate-pair" select="$cpair"/>
            </xsl:call-template>
        </xsl:variable>
        <!-- Now the same recursion as in the example. -->
        <xsl:variable name="rest" select="substring-after($wlist,' ') "/>
        <xsl:choose>
            <xsl:when test="string-length($rest) > 0">
                <xsl:choose>
                    <!-- the template is called recursively with the either the parameter passed to it
                or the current x value should this be lower -->
                    <xsl:when test="$x > $cval">
                        <xsl:call-template name="get-min-x">
                            <xsl:with-param name="list" select="$rest"/>
                            <xsl:with-param name="cval" select="$cval"/>
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:call-template name="get-min-x">
                            <xsl:with-param name="list" select="$rest"/>
                            <xsl:with-param name="cval" select="$x"/>
                        </xsl:call-template>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="$cval - 5"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
<!=============================================================================
  get-min-y: gets the minimum y (screen-)coordinate of a list of coordinates.
  Parameters:  list      space-separated list of coordinate pairs x,y
               cval      used for recursion, must be initialized to a big value
==============================================================================>
    <xsl:template name="get-min-y">
        <xsl:param name="list"/>
        <xsl:param name="cval"/>
        <xsl:variable name="wlist" select="normalize-space($list)"/>
        <xsl:variable name="cpair" select="substring-before($wlist,' ')"/>
        <xsl:variable name="y">
            <xsl:call-template name="transform-y">
                <xsl:with-param name="coordinate-pair" select="$cpair"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="rest" select="substring-after($wlist,' ') "/>
        <xsl:choose>
            <xsl:when test="string-length($rest) > 0">
```

```
                <xsl:choose>
                    <xsl:when test="$y > $cval">
                        <xsl:call-template name="get-min-y">
                            <xsl:with-param name="list" select="$rest"/>
                            <xsl:with-param name="cval" select="$cval"/>
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:call-template name="get-min-y">
                            <xsl:with-param name="list" select="$rest"/>
                            <xsl:with-param name="cval" select="$y"/>
                        </xsl:call-template>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="$cval - 5"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
<!-- ====================================================================================
    get-max-x: gets the maximum x (screen-)coordinate of a list of coordinates.
    Parameters:  list      space-separated list of coordinate pairs x,y
                 cval      used for recursion, must be initialized to a small value (0)
==================================================================================== -->
    <xsl:template name="get-max-x">
        <xsl:param name="list"/>
        <xsl:param name="cval"/>
        <xsl:variable name="wlist" select="normalize-space($list)"/>
        <xsl:variable name="cpair" select="substring-before($wlist,' ')"/>
        <xsl:variable name="x">
            <xsl:call-template name="transform-x">
                <xsl:with-param name="coordinate-pair" select="$cpair"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="rest" select="substring-after($wlist,' ') "/>
        <xsl:choose>
            <xsl:when test="string-length($rest) > 0">
                <xsl:choose>
                    <xsl:when test="$x > $cval">
                        <xsl:call-template name="get-max-x">
                            <xsl:with-param name="list" select="$rest"/>
                            <xsl:with-param name="cval" select="$x"/>
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:call-template name="get-max-x">
                            <xsl:with-param name="list" select="$rest"/>
                            <xsl:with-param name="cval" select="$cval"/>
                        </xsl:call-template>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="$cval + 5"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
<!-- ====================================================================================
    get-max-y gets the maximum x (screen-)coordinate of a list of coordinates.
    Parameters:  list      space-separated list of coordinate pairs x,y.
                 cval      used for recursion, must be initialized to a small value (0)
==================================================================================== -->
    <xsl:template name="get-max-y">
        <xsl:param name="list"/>
        <xsl:param name="cval"/>
        <xsl:variable name="wlist" select="normalize-space($list)"/>
        <xsl:variable name="cpair" select="substring-before($wlist,' ')"/>
        <xsl:variable name="y">
            <xsl:call-template name="transform-y">
                <xsl:with-param name="coordinate-pair" select="$cpair"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="rest" select="substring-after($wlist,' ') "/>
        <xsl:choose>
```

```
            <xsl:when test="string-length($rest) > 0">
                <xsl:choose>
                    <xsl:when test="$y > $cval">
                        <xsl:call-template name="get-max-y">
                            <xsl:with-param name="list" select="$rest"/>
                            <xsl:with-param name="cval" select="$y"/>
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:call-template name="get-max-y">
                            <xsl:with-param name="list" select="$rest"/>
                            <xsl:with-param name="cval" select="$cval"/>
                        </xsl:call-template>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="$cval + 5"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
<!--====\\\=========================================================================
  Template Feature: draws the text attributes associated with each parcel.
==========================================================================-->
    <xsl:template match="Feature">
        <!-- First we store the x and y coordinate of the center of gravity of the points as an
      approximation of the center of the polygon.-->
        <xsl:variable name="x">
            <xsl:call-template name="center-of-gravity-x">
                <xsl:with-param name="list"
select="./geometricProperty/Polygon/outerBoundaryIs/LinearRing/coordinates"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="y">
            <xsl:call-template name="center-of-gravity-y">
                <xsl:with-param name="list"
select="./geometricProperty/Polygon/outerBoundaryIs/LinearRing/coordinates"/>
            </xsl:call-template>
        </xsl:variable>
        <!-- match a template to position the name of the parcel just above the center -->
        <xsl:apply-templates select="./name">
            <xsl:with-param name="x" select="$x"/>
            <xsl:with-param name="y" select="$y - 5"/>
        </xsl:apply-templates>
        <!-- match the property elements, but only those with typename attribute set to owner -->
        <xsl:apply-templates select="./property[@typeName = 'BUILDING']">
            <xsl:with-param name="x" select="$x"/>
            <xsl:with-param name="y" select="$y + 10"/>
        </xsl:apply-templates>
        <!-- match the property elements, this time for the EZ's -->
        <xsl:apply-templates select="./property[@typeName = 'USAGE']">
            <xsl:with-param name="x" select="$x"/>
            <xsl:with-param name="y" select="$y + 20"/>
        </xsl:apply-templates>
    </xsl:template>
<!--=========================================================================
  Template property: Draws additional text attributes
==========================================================================-->
    <xsl:template match="property">
        <xsl:param name="x"/>
        <xsl:param name="y"/>
        <xsl:element name="text" namespace="">
            <xsl:attribute name="x"><xsl:value-of select="$x"/></xsl:attribute>
            <xsl:attribute name="y"><xsl:value-of select="$y"/></xsl:attribute>
            <xsl:attribute name="style">font-family:Verdana;font-size:12;fill:blue</xsl:attribute>
            <xsl:attribute name="text-anchor">middle</xsl:attribute>
            <xsl:value-of select="normalize-space(.)"/>
        </xsl:element>
    </xsl:template>
<!--=========================================================================
  Template LinearRing: Draws the border of the parcels
==========================================================================-->
    <xsl:template match="LinearRing">
        <xsl:variable name="clist" select="./coordinates"/>
        <xsl:variable name="tclist" select="normalize-space($clist)"/>
```

```
            <xsl:variable name="cpair" select="substring-before($tclist,' ')"/>
            <!-- Again we transform from real world to screen coordinates other than that it is pretty
          much the same as in the example. -->
            <xsl:variable name="x">
                <xsl:call-template name="transform-x">
                    <xsl:with-param name="coordinate-pair" select="$cpair"/>
                </xsl:call-template>
            </xsl:variable>
            <xsl:variable name="y">
                <xsl:call-template name="transform-y">
                    <xsl:with-param name="coordinate-pair" select="$cpair"/>
                </xsl:call-template>
            </xsl:variable>
            <xsl:variable name="start" select="concat( 'M ', $x, ' ', $y)"/>
            <xsl:variable name="rest" select="substring-after($tclist,' ')"/>
            <xsl:variable name="all">
                <xsl:call-template name="gmlcoords-to-svgpath">
                    <xsl:with-param name="list" select="$rest"/>
                </xsl:call-template>
            </xsl:variable>
            <xsl:variable name="all2" select="concat(concat($start,$all),' Z')"/>
            <xsl:element name="path">
                <xsl:attribute name="style">stroke:black;fill:none;stroke-width:.5;</xsl:attribute>
                <xsl:attribute name="d"><xsl:value-of select="$all2"/></xsl:attribute>
            </xsl:element>
        </xsl:template>
    <!--=========================================================================================
        gmlcoords-to-svgpath: Template from the example slightly modified
                    (added the coordinate transformation)
    ========================================================================================== -->
        <xsl:template name="gmlcoords-to-svgpath">
            <xsl:param name="list"/>
            <xsl:variable name="wlist" select="normalize-space($list)"/>
            <xsl:variable name="cpair" select="substring-before($wlist,' ')"/>
            <xsl:variable name="x">
                <xsl:call-template name="transform-x">
                    <xsl:with-param name="coordinate-pair" select="$cpair"/>
                </xsl:call-template>
            </xsl:variable>
            <xsl:variable name="y">
                <xsl:call-template name="transform-y">
                    <xsl:with-param name="coordinate-pair" select="$cpair"/>
                </xsl:call-template>
            </xsl:variable>
            <xsl:variable name="rest" select="substring-after($wlist,' ') "/>
            <xsl:choose>
                <xsl:when test="string-length($rest) > 0">
                    <xsl:variable name="all">
                        <xsl:call-template name="gmlcoords-to-svgpath">
                            <xsl:with-param name="list" select="$rest"/>
                        </xsl:call-template>
                    </xsl:variable>
                    <xsl:value-of select="concat(' L ', $x,' ', $y, $all)"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:variable name="lx">
                        <xsl:call-template name="transform-x">
                            <xsl:with-param name="coordinate-pair" select="$wlist"/>
                        </xsl:call-template>
                    </xsl:variable>
                    <xsl:variable name="ly">
                        <xsl:call-template name="transform-y">
                            <xsl:with-param name="coordinate-pair" select="$wlist"/>
                        </xsl:call-template>
                    </xsl:variable>
                    <xsl:value-of select="concat(' L ', $lx,' ', $ly)"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:template>
    <!--=========================================================================================
        Parameters: list        space separated list of coordinate-pairs x,y
                accX        used for recursion; accumulated sum of x values (default = 0)
                accY        used for recursion; accumulated sum of y values (default = 0)
                numb        used for recursion; number of coordinates so far (default = 0)
    ========================================================================================== -->
```

```
    <xsl:template name="center-of-gravity-x">
        <xsl:param name="list"/>
        <xsl:param name="accX" select="0"/>
        <xsl:param name="accY" select="0"/>
        <xsl:param name="numb" select="0"/>
        <xsl:variable name="wlist" select="normalize-space($list)"/>
        <xsl:variable name="cpair" select="substring-before($wlist,' ')"/>
        <!-- works like that: x, y screen coordinates are stored -->
        <xsl:variable name="x">
            <xsl:call-template name="transform-x">
                <xsl:with-param name="coordinate-pair" select="$cpair"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="y">
            <xsl:call-template name="transform-y">
                <xsl:with-param name="coordinate-pair" select="$cpair"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="rest" select="substring-after($wlist,' ') "/>
        <xsl:choose>
            <!-- template is called recursively, adding the current to the accumulated values -->
            <xsl:when test="string-length($rest) > 0">
                <xsl:call-template name="center-of-gravity-x">
                    <xsl:with-param name="list" select="$rest"/>
                    <xsl:with-param name="accX" select="$accX + $x"/>
                    <xsl:with-param name="accY" select="$accY + $y"/>
                    <xsl:with-param name="numb" select="$numb + 1"/>
                </xsl:call-template>
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="round($accX div $numb)"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
<!-- ================================================================================
    center-of-gravity-y, calculates the center of gravity of a set of points. The same remark as
      center-of-gravity-x applies.
    Parameters: list        space separated list of coordinate-pairs x,y
            accX        used for recursion; accumulated sum of x values (default = 0)
            accY        used for recursion; accumulated sum of y values (default = 0)
            numb        used for recursion; number of coordinates so far (default = 0)
================================================================================ -->
    <xsl:template name="center-of-gravity-y">
        <xsl:param name="list"/>
        <xsl:param name="accX" select="0"/>
        <xsl:param name="accY" select="0"/>
        <xsl:param name="numb" select="0"/>
        <xsl:variable name="wlist" select="normalize-space($list)"/>
        <xsl:variable name="cpair" select="substring-before($wlist,' ')"/>
        <xsl:variable name="x">
            <xsl:call-template name="transform-x">
                <xsl:with-param name="coordinate-pair" select="$cpair"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="y">
            <xsl:call-template name="transform-y">
                <xsl:with-param name="coordinate-pair" select="$cpair"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="rest" select="substring-after($wlist,' ') "/>
        <xsl:choose>
            <xsl:when test="string-length($rest) > 0">
                <xsl:call-template name="center-of-gravity-y">
                    <xsl:with-param name="list" select="$rest"/>
                    <xsl:with-param name="accX" select="$accX + $x"/>
                    <xsl:with-param name="accY" select="$accY + $y"/>
                    <xsl:with-param name="numb" select="$numb + 1"/>
                </xsl:call-template>
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="round($accY div $numb)"/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
<!-- ================================================================================
```

```
    Template coordinates: used to mark points
=================================================================================================>
    <xsl:template match="coordinates">
        <xsl:call-template name="mark-points">
            <xsl:with-param name="list" select="."/>
        </xsl:call-template>
    </xsl:template>


<!=================================================================================================
    mark-points: marks point with circles
    Parameters: list      space separated list of coordinate-pairs x,y
=================================================================================================>
    <xsl:template name="mark-points">
        <xsl:param name="list"/>
        <xsl:variable name="wlist" select="normalize-space($list)"/>
        <xsl:variable name="cpair" select="substring-before($wlist,' ')"/>
        <xsl:variable name="x">
            <xsl:call-template name="transform-x">
                <xsl:with-param name="coordinate-pair" select="$cpair"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="y">
            <xsl:call-template name="transform-y">
                <xsl:with-param name="coordinate-pair" select="$cpair"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="rest" select="substring-after($wlist,' ') "/>
        <xsl:choose>
            <xsl:when test="string-length($rest) > 0">
                <xsl:call-template name="mark-points">
                    <xsl:with-param name="list" select="$rest"/>
                </xsl:call-template>
                <xsl:element name="circle" namespace="">
                    <xsl:attribute name="cx"><xsl:value-of select="$x"/></xsl:attribute>
                    <xsl:attribute name="cy"><xsl:value-of select="$y"/></xsl:attribute>
                    <xsl:attribute name="r">1</xsl:attribute>
                    <xsl:attribute name="style">fill:red;stroke:red;stroke-width:.10;</xsl:attribute>
                </xsl:element>
            </xsl:when>
            <xsl:otherwise/>
        </xsl:choose>
    </xsl:template>
</xsl:stylesheet>
```

APPENDIX 11 LINE.XSL

```
<!--    Project                 : Integrated XML and GML in Geography Information System
        Developer               : Mardiana Binti Abdul Rahman
        ID                      : 1809
        Institution             : University Technologi PETRONAS
        Document Description    : This document consist of eXtensible Stylesheet Language
                                  Transformation(XSLT). This stylesheet will transform the GML
                                  into SVG and will be viewed in web browser -->

<!-- SCHEMA DECLARATION -->

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
xmlns:saxon="http://icl.com/saxon" xmlns:Extfun="/org.opengis.gml.StyleExt"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="http://www.opengis.net/gml"
xmlns:gml.tm="http://www.opengis.net/gml/temporal" xmlns:wfs="http://www.galdos.com/wfs"
xmlns:exp="http://www.opengis.net/examples">

<!-- DEFINE BOUNDING BOX ELEMENT -->

        <xsl:output indent="yes"/>
        <xsl:param name="CONVTYPE">pixel</xsl:param>
        <xsl:param name="WIDTH">500</xsl:param>
        <xsl:param name="HEIGHT">400</xsl:param>

<!-- BODY -->

        <xsl:variable name="sumY">
                <xsl:apply-templates select="exp:FeatureCollection/gml:boundedBy/gml:Box/gml:coordinates"
mode="sumY"/>
        </xsl:variable>
        <xsl:variable name="scaleX">
                <xsl:apply-templates select="exp:FeatureCollection/gml:boundedBy/gml:Box/gml:coordinates"
mode="scaleX"/>
        </xsl:variable>
        <xsl:variable name="scaleY">
                <xsl:apply-templates select="exp:FeatureCollection/gml:boundedBy/gml:Box/gml:coordinates"
mode="scaleY"/>
        </xsl:variable>
        <xsl:variable name="x1">
                <xsl:apply-templates select="exp:FeatureCollection/gml:boundedBy/gml:Box/gml:coordinates"
mode="x1"/>
        </xsl:variable>
        <xsl:variable name="y1">
                <xsl:apply-templates select="exp:FeatureCollection/gml:boundedBy/gml:Box/gml:coordinates"
mode="y1"/>
        </xsl:variable>
        <xsl:variable name="minus">-1</xsl:variable>
        <xsl:variable name="one">1</xsl:variable>
        <xsl:variable name="inverseScale">
                <xsl:value-of select="$one div $scaleX"/>
        </xsl:variable>
        <xsl:template match="/">
                <xsl:element name="svg">
                        <xsl:attribute name="width"><xsl:value-of select="$WIDTH"/></xsl:attribute>
                        <xsl:attribute name="height"><xsl:value-of select="$HEIGHT"/></xsl:attribute>
                        <xsl:element name="defs">
                                <xsl:element name="g">
                                        <xsl:attribute name="id">Lib1_9</xsl:attribute>
                                        <xsl:if test="$CONVTYPE='pixel'">
                                                <xsl:attribute
name="transform">scale(0.25)</xsl:attribute>
                                        </xsl:if>
                                        <xsl:if test="$CONVTYPE='user'">
                                                <xsl:attribute name="transform">scale(<xsl:value-of
select="$inverseScale"/>) scale(0.25) matrix(1 0 0 -1 0 70)</xsl:attribute>
                                        </xsl:if>
                                        <xsl:element name="path">
                                                <xsl:attribute name="style">fill-
rule:nonzero;fill:#EBF322;stroke:#000000;stroke-miterlimit:4;</xsl:attribute>
                                                <xsl:attribute
name="d">M68.262,74.51H0.5V37.255h67.762V74.51z</xsl:attribute>
                                        </xsl:element>
```

```
                                                <xsl:element name="path">
                                                        <xsl:attribute name="style">fill-
rule:nonzero;fill:#EBF322;stroke:#000000;stroke-miterlimit:4;</xsl:attribute>
                                                        <xsl:attribute
name="d">M13.793,17.647h41.176</xsl:attribute>
                                                </xsl:element>
                                                <xsl:element name="path">
                                                        <xsl:attribute name="style">fill-
rule:nonzero;fill:#EBF322;stroke:#000000;stroke-miterlimit:4;</xsl:attribute>
                                                        <xsl:attribute
name="d">M31.44,0v35.294</xsl:attribute>
                                                </xsl:element>
                                        </xsl:element>
                                </xsl:element>
                                <xsl:if test="$CONVTYPE = 'pixel'">
                                        <xsl:apply-templates select="//exp:RL1U/gml:centerLineOf"
mode="Line"/>
                                </xsl:if>
                                <xsl:if test="$CONVTYPE = 'user'">
                                        <xsl:element name="g">
                                                <xsl:attribute name="transform">scale(<xsl:value-of
select="$scaleX"/>,<xsl:value-of select="$scaleY"/>) translate(<xsl:value-of select="$minus*$x1"/>,<xsl:value-of
select="$minus*$y1"/>) matrix(1 0 0 -1 0 <xsl:apply-templates select="/gml:boundedBy/gml:Box/gml:coordinates"
mode="sumY"/>)</xsl:attribute>
                                                <xsl:apply-templates select="//exp:RL1U/gml:centerLineOf"
mode="Line"/>
                                        </xsl:element>
                                </xsl:if>
                        </xsl:element>
                </xsl:template>
                <xsl:template match="gml:boundedBy/gml:Box/gml:coordinates" mode="scaleX">
                        <xsl:value-of
select="Extfun:getScaleX(string($WIDTH),string(text()),string(@decimal),string(@cs),string(@ts))"/>
                </xsl:template>
                <xsl:template match="gml:boundedBy/gml:Box/gml:coordinates" mode="scaleY">
                        <xsl:value-of
select="Extfun:getScaleY(string($HEIGHT),string(text()),string(@decimal),string(@cs),string(@ts))"/>
                </xsl:template>
                <xsl:template match="gml:boundedBy/gml:Box/gml:coordinates" mode="x1">
                        <xsl:value-of select="Extfun:getX1(string(text()),string(@decimal),string(@cs),string(@ts))"/>
                </xsl:template>
                <xsl:template match="gml:boundedBy/gml:Box/gml:coordinates" mode="y1">
                        <xsl:value-of select="Extfun:getY1(string(text()),string(@decimal),string(@cs),string(@ts))"/>
                </xsl:template>
                <xsl:template match="gml:boundedBy/gml:Box/gml:coordinates" mode="sumY">
                        <xsl:value-of
select="Extfun:getsumY(string(text()),string(@decimal),string(@cs),string(@ts))"/>
                </xsl:template>
                <xsl:template match="gml:boundedBy/gml:Box/gml:coordinates" mode="inverseScale">
                        <xsl:value-of
select="Extfun:getInverseScale(string(text()),string(@decimal),string(@cs),string(@ts))"/>
                </xsl:template>
                <xsl:template match="exp:RL1U/gml:centerLineOf" mode="Line">
                        <xsl:variable name="strokeWidth">
                                <xsl:value-of select="1"/>
                        </xsl:variable>
                        <xsl:variable name="rStroke">
                                <xsl:value-of select="255"/>
                        </xsl:variable>
                        <xsl:variable name="gStroke">
                                <xsl:value-of select="0"/>
                        </xsl:variable>
                        <xsl:variable name="bStroke">
                                <xsl:value-of select="0"/>
                        </xsl:variable>
                        <xsl:variable name="idVar">
                                <xsl:call-template name="getAnc">
                                        <xsl:with-param name="featureName" select="'exp:RL1U'"/>
                                </xsl:call-template>
                        </xsl:variable>
                        <xsl:element name="path">
                                <xsl:if test="$CONVTYPE='pixel'">
                                        <xsl:attribute name="style">stroke-width:<xsl:value-of
select="$strokeWidth"/>;fill:none;stroke:rgb(<xsl:value-of select="$rStroke"/>,<xsl:value-of
select="$gStroke"/>,<xsl:value-of select="$bStroke"/>);</xsl:attribute>
```

```
                        </xsl:if>
                        <xsl:if test="$CONVTYPE='user'">
                                <xsl:attribute name="style">stroke-width:<xsl:value-of
select="$strokeWidth*$inverseScale"/>;fill:none;stroke:rgb(<xsl:value-of select="$rStroke"/>,<xsl:value-of
select="$gStroke"/>,<xsl:value-of select="$bStroke"/>);</xsl:attribute>
                        </xsl:if>
                        <xsl:attribute name="id"><xsl:value-of select="$idVar"/></xsl:attribute>
                        <xsl:attribute name="d"><xsl:apply-templates select=".//gml:coordinates"
mode="Path"/></xsl:attribute>
                </xsl:element>
        </xsl:template>

        <xsl:template name="getAnc">
                <xsl:param name="featureName"/>
                <xsl:choose>
                        <xsl:when test="name(.) = $featureName">
                                <xsl:value-of select="./@ID"/>
                        </xsl:when>
                        <xsl:otherwise>
                                <xsl:for-each select="..">
                                        <xsl:call-template name="getAnc">
                                                <xsl:with-param name="featureName"
select="$featureName"/>
                                        </xsl:call-template>
                                </xsl:for-each>
                        </xsl:otherwise>
                </xsl:choose>
        </xsl:template>
</xsl:transform>

<!-- END -->
```