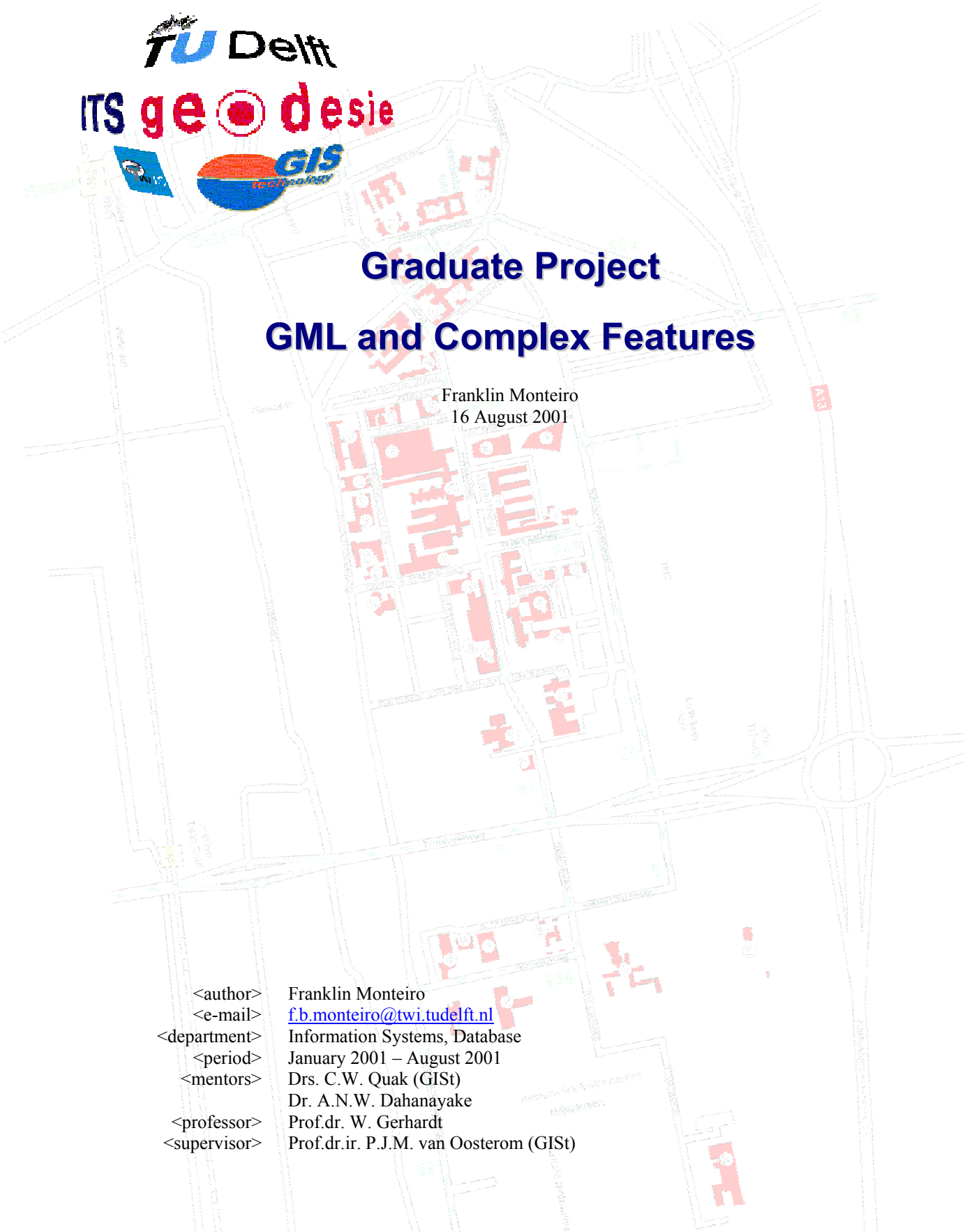


Graduate Project

GML and Complex Features

Franklin Monteiro
16 August 2001

<author> Franklin Monteiro
<e-mail> f.b.monteiro@twi.tudelft.nl
<department> Information Systems, Database
<period> January 2001 – August 2001
<mentors> Drs. C.W. Quak (GISt)
Dr. A.N.W. Dahanayake
<professor> Prof.dr. W. Gerhardt
<supervisor> Prof.dr.ir. P.J.M. van Oosterom (GISt)

Preface

This report is the result of the graduate project “GML and Complex Features” which has been performed at the department of Geodesy, section GIS-technology (GIS-t) of the faculty of Civil Engineering and Geosciences, in the final stage of the study Technical Informatics (specialization department Information Systems, section Databases) at the Delft University of Technology.

The author offers special thanks to the supervisors, co-workers of the section GIS-t and to others who have contributed to the realization of this graduation. A special word of thanks in particular to A. Dahanayake, W. Quak, W. Gerhardt, P. van Oosterom and T. Tijssen.

Abstract

Geographic Information Systems (GIS) are information systems that centre on geographic data, that is data related to a location on earth. In GISs geographic information usually is abstracted to features (geographic classes). The features are the units of geographic information.

Developments in the GIS technology have indicated that connections between the GISs are required and a network of GISs over the internet is considered. In order to achieve this, it is necessary to develop standards. The consortium OpenGIS® is engaged in the development of such a standard and produced a series of specifications for that purpose, of which one is the Geography Markup Language (GML) specification. GML is an XML vocabulary, meant to encode geographic data for storage and transport (within a GIS network). XML (eXtensible Markup Language) is a standard of the W3C to structure data. An XML vocabulary restricts XML documents to a certain structure.

The current version of the GML specification, GML 2.0, is based on the Simple Features Model: a simplified implementation model of the feature model of OpenGIS. At the same time the specification indicates that extension with non-simple features, or complex features, is necessary. The purpose of this project is to realize this extension.

Study of literature preceding this graduation has shown that complex in this context is analogous to topology: Complex spatial values are sets of spatial values geographically linked to each other. Complex features are sets of features geographically linked to each other. Models are available for (describing) topology. The winged-edge model is chosen (in the preliminary research) as starting point for extending GML with topology.

The GML specification includes UML models and XML schemas that are formed on the basis of these UML models. The extension consists of replacing UML models and XML schemas. The winged-edge principle is modelled in UML and with that, a number of alternative UML models are formed, to replace the original GML models. Finally one of those is elaborated and translated to corresponding XML schemas.

The usefulness of the extended GML is tested by generating GML from existing data, which are loaded into a database for this purpose. Afterwards a Java GIS application (QuickGIS) is adjusted to visualize GML files.

Content

PREFACE ..... **I**

ABSTRACT ..... **II**

CONTENT ..... **III**

FIGURES ..... **V**

TABLES V

1 INTRODUCTION **1**

2 TASK DESCRIPTION **2**

2.1 GIS PROJECT 2
2.2 PROJECT OF GRADUATION 2

3 OVERVIEW **3**

3.1 STANDARDS 3
3.2 SPATIAL VALUES 4
3.3 FEATURES 5
3.4 TOPOLOGIC MODELS 6
3.5 XML 8

4 CURRENT SITUATION: GML 2.0 ..... **11**

4.1 STRUCTURE 11
4.2 GEOMETRY XSD 11
4.3 FEATURE XSD 12
4.4 APPLICATION SPECIFIC XSDS 12

5 REQUIRED SITUATION **13**

5.1 GEOMETRY 13
5.2 TOPOLOGY 13

6 ALTERNATIVES **14**

6.1 COMPLEX 14
6.2 FEATURE 15
6.3 LOCATION COMPLEX 16

7 RECOMMENDED CHOICE **17**

8 FURTHER ELABORATION **18**

8.1 COMPOSITE 18
8.2 COMPLEXES HIERARCHY 18
8.3 COMPLEX FEATURES 18
8.4 COMPLETE MODEL 18
8.5 XSD SCHEMAS 18

9 IMPLEMENTATION **20**

9.1 PREPARATION: LOAD TOPOLOGIC DATA IN DATABASE 20
9.2 GENERATE GML FROM DATABASE 22
9.3 VISUALIZE GML 22

10 CONCLUSIONS, RECOMMENDATIONS **24**

REFERENCES **25**

SOFTWARE 26

APPENDIX A XML SPECIFICATIONS **27**

A.1 XML 27
A.2 XML SCHEMA 29
A.3 XLINK 30

APPENDIX B GML EXAMPLES 31

B.1 EXAMPLE 1: GEOMETRYCOMPLEX 31
B.2 EXAMPLE 2: STATE 31
B.3 EXAMPLE 3: FACGEO 33

APPENDIX C SCHEMAS 38

C.1 ENCODING GML++ 38
C.2 GEOMETRY.XSD 38
C.3 FEATURE.XSD 40

APPENDIX D SOURCE CODE 43

D.1 SCRIPTS 43
D.2 GML 47
D.3 QUICKGIS 53

Figures

FIGURE 1 CLASS HIERARCHY GEOMETRIC OBJECT [14] 5
FIGURE 2 CLASS HIERARCHY TOPOLOGIC OBJECT [14] 5
FIGURE 3 RELATIONS WITHIN TOPOLOGIC OBJECT [14] 5
FIGURE 4 RELATION BETWEEN GEOMETRIC EN TOPOLOGIC OBJECTS [14] 5
FIGURE 5 ABSTRACT FEATURE MODEL (OPENGIS) [15] 6
FIGURE 6 GENERAL FEATURE MODEL (ISO /TC211) [15] 6
FIGURE 7 GENERAL SINGLE-VALUED VECTOR MAPS [8] 7
FIGURE 8 SIMPLIFIED GENERAL SINGLE-VALUED VECTOR MAPS [8] 8
FIGURE 9 WINGED-EDGE TOPOLOGY [3] 8
FIGURE 10 XML ELEMENTS 9
FIGURE 11 DOM TREE 10
FIGURE 12 GML GEOMETRIC MODEL 12
FIGURE 13 GML FEATURE MODEL 12
FIGURE 14 EXAMPLE XSD EN GML FRAGMENT 12
FIGURE 15 GEOMETRY 13
FIGURE 16 CHAIN TOPOLOGY 13
FIGURE 17 OPTION A: TOPOLOGY CONTAINS GEOMETRY 14
FIGURE 18 OPTION B: TOPOLOGY AS AN ASSOCIATION CLASS 14
FIGURE 19 OPTION C: TOPOLOGY WITHIN GEOMETRY 14
FIGURE 20 OPTION D: TOPOLOGY SEPARATE FROM GEOMETRY 14
FIGURE 21 OPTION I: TOPOLOGY ON THE FEATURE LEVEL 15
FIGURE 22 OPTION II: PROPERTIES FOR TOPOLOGIC PRIMITIVES 15
FIGURE 23 OPTION III: PROPERTY FOR COMPLEXES 15
FIGURE 24 6.2.4 OPTION IV: PRIMITIVES CONTAIN FEATURES 15
FIGURE 25 OPTION V: FORMER SITUATION 15
FIGURE 26 COMPOSITE 18
FIGURE 27 COMPLEX FEATURE 18
FIGURE 28 NEW GEOMETRY MODEL 18
FIGURE 29 NEW FEATURE MODEL 18
FIGURE 30 ARCHITECTURE 20
FIGURE 31 SCREENSHOT ARCEDIT 20
FIGURE 32 ARC2ORA.SH SCRIPT 20
FIGURE 33 SCRIPTS 20
FIGURE 34 DATAMODEL OF THE TESTDATA LOADED IN ORACLE 21
FIGURE 35 QUICKGIS SREENSHOT 23

tables

TABLE 1 ISO TC 211 STANDARDS 3
TABLE 2 OPENGIS SPECIFICATIONS 4
TABLE 3 GML GEOMETRIC ELEMENTS 11
TABLE 4 GML PROPERTY ELEMENTS 12
TABLE 5 XSD SIMPLE TYPES GROUPED BY SORT 29
TABLE 6 IMPORTANT XSD ELEMENTS 30
TABLE 7 XLINK ATTRIBUTES 30

1 Introduction

GML (Geography Markup Language) is an XML encoded language for transport and storage of geographic information. The GML specification defines mechanisms and syntax for the description of geographic information in XML. The described formats serve as a storage format as well as an exchange format for geographic files. The GML 2.0 specification [13] has been written by the OpenGIS Consortium. The current version of the specification merely describes the encoding of simple geometric objects (points, lines, polygons). Within the world of GIS however, methods to describe complex geometric objects are also required. On the other hand the abstract specification of the OpenGIS do describe complex objects in accordance with the ISO model [7].

The intention is to expand GML to such an extent that it can contain complex features.

In chapter 2 the task will be described extensively. Chapter 3 offers an overview (background information) of the subjects and terms that come up for discussion. Chapter 4 describes the current GML specification, whereas in the chapters 5 to 8 a new GML model is developed: a number of alternatives are contemplated (chapter 6), of which one is chosen (chapter 7) to be elaborated (chapter 8).

Subsequently the implementation is discussed in chapter 9 and the conclusions and recommendations come up for discussion in chapter 10. In appendix A another few XML specifications are described. In appendix B examples of GML are included. The new GML schemas are included in appendix C and appendix D contains the implemented source code.

2 Task description

2.1 GIS Project

This graduate project is part of a coordinating (internet-) GIS research of the GIS-technology section of the faculty of Civil Engineering and Geosciences (Delft University of Technology), department of Geodesy. Next, a detailed description of some of the aspects of this research is given.

2.1.1 DBMS

At the basis of a GIS is the spatial database. A spatial database system is a database system with additionally the means to work with spatial data. At the very least this is the possibility to include spatial data types in the data model and in the query language and efficient algorithms for spatial database operations and spatial indexing [4]. Ideally these functions are realized by the DBMS. However the possibilities of the present generation of these Geo-DBMSs are still limited. New developments on the area of Geo-DBMSs are an important part of the research program [10].

2.1.2 Internet GIS

The main idea is to form a distributed network of GISs (Geo-Information Infrastructure [10]) connected over the internet. Such a GIS can be approached and inquired through the internet. Combining geographic information of different location will have to be possible in this. GML could play an important role as exchange format.

2.1.3 Magma/Lava

At the moment in the department of Geodesy of the Delft University of Technology research is being done on the Magma/Lava system [29]. Lava is a (Java) GIS browser, with the ability to communicate with several Magma servers through the internet and to show the collected map information. A Magma server translates a request of a Lava client to a SQL query for the underlying relational database. In the current version Magma and Lava communicate with a self-invented protocol. At the moment a version in which these two programs will use GML to communicate, is being developed. A version of Magma able to work with a database according to the Oracle object model is also being developed. Another possibility that is considered is to have Magma communicate with other databases (like the object oriented Perspective-DB [24]).

2.2 Project of graduation

GML (Geography Markup Language) is an XML encoded language for transport and storage of geographic information. The GML specification defines the mechanisms and syntax to describe geographic information in XML. The described formats serve as storage as well as exchange format for geographic files. The GML 2.0 specification [13] has been written by the OpenGIS Consortium (www.opengis.org). The current version of the specification merely describes the encoding of simple geometric objects (points, curves, surfaces). Within the world of GIS however, methods to describe complex geometric objects are also required. On the other hand the abstract specifications of OpenGIS do describe complex objects.

The aim of this graduate project is to extend GML specification with complex objects. Precedingly, the definitions and models of complex objects and features have been determined through study of literature. These obtained insights serve as a starting point for composing a new GML model and specification.

The activities in the graduate project are:

- Reading about the matter; XML, GML, OpenGIS specifications of simple and complex objects. At Geodesy the XML parser software Xerces [23] is installed. The package Xalan [22] for the processing of style sheets is also present.
- Writing a GML specification for complex objects
- The implementation of the whole within Magma/Lava will be regarded in collaboration with other students, after which experimenting is possible. A part of all this is the loading of complex objects in the Oracle database.
- Finding a suitable testdataset

3 Overview

This chapter offers an overview of concepts playing a role in this project. Paragraph 3.1 offers an overview of the standards on the field of GIS. In 3.2 and 3.3 the concepts spatial values and features from these standards are illustrated. In 3.4 the topologic models are explained. Finally the basics of XML are discussed in 3.5.

3.1 Standards

Standardization is needed for the exchange of data between GISs or to connect GISs. A number of standardization organisations are occupied with this, including ISO and OpenGIS. These two organisations work together to attune their standards to each other.

3.1.1 ISO TC 211

At ISO this is the responsibility of the working group TC 211. This working group has created a family of standards under the project numbers 19101 to 19128 (formerly numbered from 15046-1 to 15046-19) (table 1). These standards formally specify methods, tools and services to acquire, process, analyse, present and migrate between different users, systems and locations [5]. During this, existing standards from the information technology are being used as much as possible.

Project nr	Title	Project nr	Title
19101 (15046-1)	Reference model	19120 (15854)	Functional standards
19102 (15046-2)	Overview	19120/Am. 1	Functional standards - Amendment 1
19103 (15046-3)	Conceptual schema language	19121 (16569)	Imagery and gridded data
19104 (15046-4)	Terminology	19122 (16822)	Qualifications and Certification of Personnel
19105 (15046-5)	Conformance and testing	19123 (17753)	Schema for coverage geometry and functions
19106 (15046-6)	Profiles	19124 (17754)	Imagery and gridded data components
19107 (15046-7)	Spatial schema	19125-1	Simple feature access - Part 1
19108 (15046-8)	Temporal schema	19125-2	Simple feature access - Part 2
19109 (15046-9)	Rules for application schema	19126	Profile - FACC Data Dictionary
19110 (15046-10)	Feature cataloguing methodology	19127	Geodetic codes and parameters
19111 (15046-11)	Spatial referencing by coordinates	19128	Web Map server interface
19112 (15046-12)	Spatial referencing by geographic identifiers		
19113 (15046-13)	Quality principles		
19114 (15046-14)	Quality evaluation procedures		
19115 (15046-15)	Metadata		
19116 (15046-16)	Positioning services		
19117 (15046-17)	Portrayal		
19118 (15046-18)	Encoding		
19119 (15046-19)	Services		
19120 (15854)	Functional standards		

table 1 ISO tc 211 standards

3.1.2 OpenGIS

The OpenGIS consortium is an initiative of the GIS industry that since 1994 has been working on a standard to solve the lack of interoperability between GISs. To the members of the consortium are reckoned amongst others: data suppliers, government authorities, hardware and software developers, universities and research institutions.

The consortium is working on Information Communities in which all imaginable GISs can simply be connected. The standards specify three models [16], viz.:

Open Geodata Model
A general data model for the basic types of geographic information and their extensions

OpenGIS Services
A client-server model for services for access to and the process of geodata (within a Information Community) and exchange of geodata (between Information Communities)

Information Communities Model
Schema that catalogues definitions and the role of and relations between elements of the open Geodata Model and the OpenGIS Services and thus realizes automated translations of geodata (between Information Communities)

The standards are subdivided in the *Abstract Specifications* and the *Implementation Specifications*, with the difference that Abstract Specifications are platform independent and the Implementation Specifications are platform specific elaborations of the Abstract Specifications (table 2). The OpenGIS Abstract Specification often sticks to the ISO standards. Meanwhile the GML specification has achieved the status of Implementation Specification.

OpenGIS® Abstract Specification		OpenGIS® Implementation Specifications
Topic nr.	*Topic Name*	Simple Features Specification for OLE/COM
Topic 0	Overview	Simple Features Specification for CORBA
Topic 1	Feature Geometry	Simple Features Specification for SQL
Topic 2	Spatial Reference Systems	Catalog Interface Implementation Specification
Topic 3	Locational Geometry	Grid Coverages Implementation Specification
Topic 4	Stored Functions and Interpolation	Coordinate Transformation Services Implementation Specification
Topic 5	Features and Feature Collections	Web Map Server Interfaces Implementation Specification
Topic 6	The Coverage Type	Geography Markup Language (GML) 2.0 Implementation Specification
Topic 7	Earth Imagery	
Topic 8	Relations Between Features	
Topic 9	Quality	**OpenGIS® Recommendations**
Topic 10	Transfer Technology	Recommended Definition Data for Coordinate Reference Systems and Coordinate Transformations 1.0.1
Topic 11	Metadata	
Topic 12	OpenGIS Service Architecture	
Topic 13	Catalogs	
Topic 14	Semantics and Information Communities	
Topic 15	Image Exploitation Services	
Topic 16	Image Coordinate Transformation Services	

table 2 OpenGIS specifications

3.2 Spatial values

Geometry has three aspects [8], viz.:

1. position and orientation
2. form and size
3. topology

The position and orientation must be given with regard to a reference system.

The topology describes the way in which parts of the underlying geometry are related (topologic relations). Mathematically seen, topology is the geometric properties that remain unchanged under continuous transformations (such as translation, scaling and rotation). The form and size are invariant to the chosen reference system. Spatial values can be divided into primitives, aggregates and complexes.

3.2.1 Primitives

The simplest spatial values are named *primitives* [7][14]. A spatial value is a primitive or is constructed of a combination of primitives. The geometric primitives in the two dimensional space are *points* (0D), *curves* (1D) and *surfaces* (2D). The topologic primitives are *nodes* (0D), *edges* (1D) and *faces* (2D). Geometric primitives are being used to model individual geometric objects in space. For example a point can represent the location of a city; a line can represent a road and a surface a lake.

3.2.2 Aggregates

A set of geometric primitives is called an *aggregate*. The geometric aggregates are being used to compose geometric objects from geometric primitives, for example a group of islands.

3.2.3 Complexes

A set of spatial values can exist of elements that are mutually related. These are so-called complex spatial values. None of the elements of a complex spatial value is supposed to overlap. The geometry of a complex value can be seen as an aggregate of all surfaces, curves (between the surfaces) and points (endpoints of the curves). This geometry is called a complex geometry. By definition a complex geometry is closed under the *boundary* operation, that means that for any primitive that is part of the complex geometry, there is a subset of all primitives (of a lower dimension) forming its boundary. The adjacency relations between the primitives form the (complex) topology of a complex geometry. In a topologic complex the spatial relations between the topologic primitives are established.

The simplest complexes are (1-dimensional) a line, from which the geometry consists of an aggregate of 1 curve and 2 points, and (2-dimensional) a triangle from which the geometry consists of an aggregate of 1 surface, 3 curves and 3 points.

A sub-complex of a complex is a subset of this complex, that individually also forms a complex. Reversibly, the larger complex is a super-complex of the other. That is why the boundary of a complex always is a sub-complex of the concerned complex.

Applications

The most important applications of complexes are (planar) partitions and (linear) networks. A partition is a division of a surface. As it were the space itself is being modelled here. Examples are topographical

maps (for example provinces as in the Netherlands), zip code areas, and land registry maps. A network (or graph) is the 1-dimensional variant. Examples are nets of highways, rivers/waterways, public transport nets and high voltage nets. The difference between networks and partitions lies in the fact that with networks the surfaces between the lines are not interesting, but the lines of connection are. On the contrary, with partitions the surfaces are important. Partitions and networks can be nested. An example is a nested partition of a land into provinces, the provinces into municipalities, the municipalities into districts, the districts into house blocks and streets.

Complexes are also being used to model geometries that in principle can be modelled by one primitive. The values of these complexes should thus be seen as primitives. An example is a line composed of straight and curved segments. The complex geometry then consists of the segments and concerning starting points and endpoints.

Modelling

It is not necessary to model a complex spatial value by describing both its geometry as well as the topology. The complex geometry is sufficient and the topology can be calculated by on the basis of the geometry. Nevertheless it is recommended to model the topology along, because of typical spatial operations on complexes, namely the topologic relations (amongst others *adjacent*, *inside* and *disjoint*). These operations are very computing-intensive without topology, whereas a suitable topologic model can lower the computing times drastically.

Advantages

Complex spatial objects can thus be modelled with aggregates just as well. The need for complex values lies in the wish to model the topology along. Advantages are:

- A suitable topologic model can efficiently calculate the otherwise very computing-intensive topologic relation operations
- The restrictions applied to the primitives with regard to each other (for example no overlap) can be warranted.
- Redundancy is prevented by a topologic model.

Model

OpenGIS uses the following data model for spatial values, as shown in figure 1 to 4. Geometry and topology are modelled as individual objects [7][14].

A geometric object, class *GM_Object*, is a primitive, an aggregate or a complex (figure 1). Composites (*GM_Composite*) are complexes that can be used as primitives. The *oriented* classes are oriented versions of the primitives. In this model only geometric curves and surfaces have an orientation.

A topologic object, class *TP_Object*, is a primitive or a complex (figure 2). The *DirectedTopo* classes serve to model oriented topologic primitives. These oriented primitives serve to realize an algebraic representation.

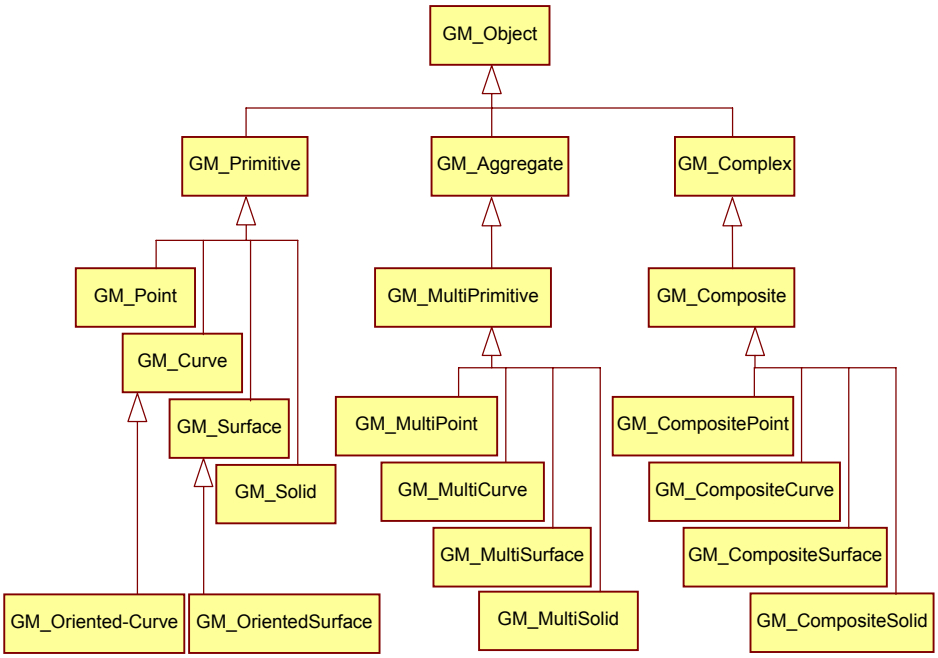


figure 1 class hierarchy geometric object [14]

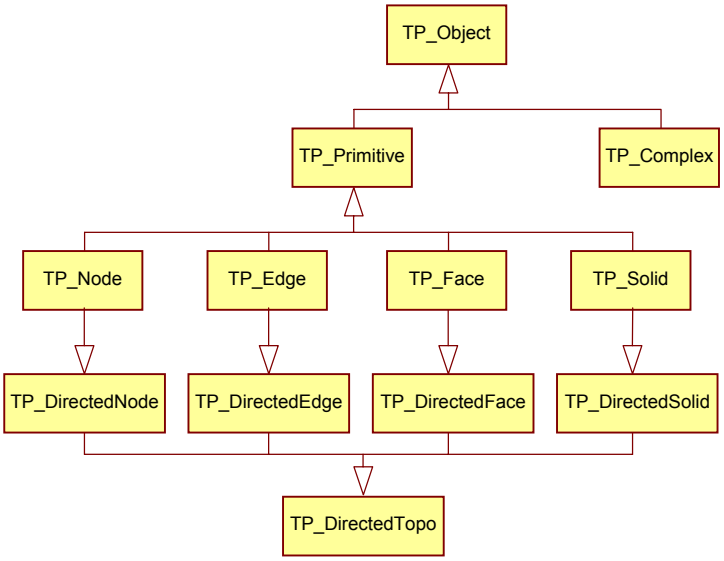


figure 2 class hierarchy topologic object [14]

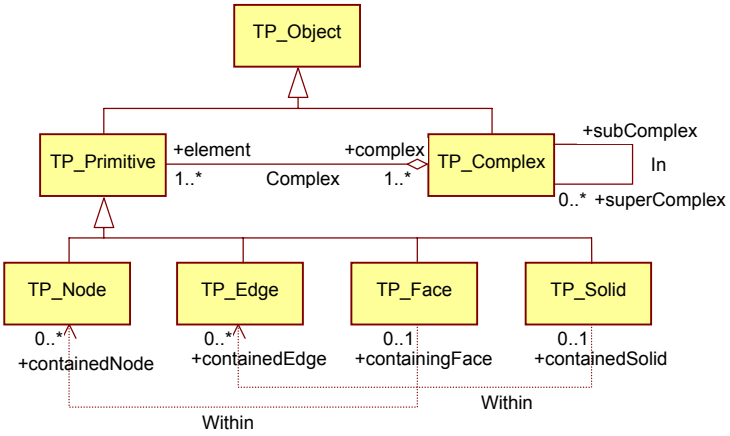


figure 3 relations within topologic object [14]

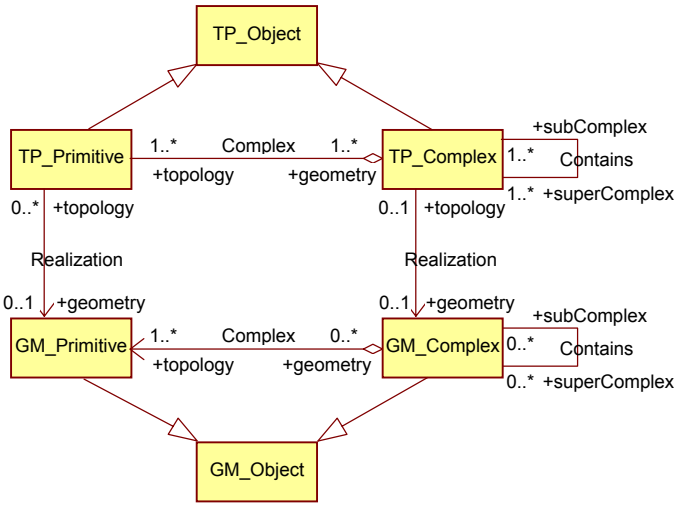


figure 4 relation between geometric en topologic objects [14]

3.3 Features

A *feature* is an abstraction of a real/existing phenomenon. The feature is the unit of geographic information. A geographic feature is a feature that is related to a location relative to the earth. Features are abstracted by contemplating certain measurable or describable characteristics (*feature attributes*) and certain procedures to manipulate features (*operators*). Geographic features contain one or more spatial characteristics. The other attributes are named thematic attributes. For the description of an instance of a feature, values must be assigned to the feature attributes.

3.3.1 Essential Model

The *Essential Model* [15] of the OGC introduces nine abstraction levels between the physical reality and features. Each level elaborates on the previous one. Starting with the physical reality those are:

Real World
The set of all facts of the physical reality.

Conceptual World
A description of the Real World in the natural language.

Geospatial World
Simplification of physical phenomenons into entities.

Dimensional World
Abstraction of the Geospatial World into quantitative relations (including geometric and positional).

Project World
A (semantic) structured part of the Dimensional World for a certain use. The OGC is working on standardization on this level. The Project World is an abstraction of the Dimensional World in terms of Feature Instances, Geometry, Corners, Geometry Schema, Spatial Reference Systems, Feature Types, Attribute-Value Pairs, Attribute Schema, and Project Schema (see figure 5). The Schemas are formal descriptions of the parts in the Project World. The Project Schema is a description of the Project World by means of metadata, so that users of the Project World are able to interpret the instances of the Features correctly. Different Project Worlds for different *Geographic Information Communities* belong with one Dimensional World.

The next four abstraction levels are mathematical and symbolic models of the physical world that are fit for software implementation. All four levels are also directly connected to with the Project World.

Point World
Describes the way in which points are defined for a Project World. In this world points are described in a Cartesian space and the corresponding coordinate systems.

Geometry World
Describes the way in which geometry should be constructed from the points to realize the Geospatial World. In this world lines and polygons are described.

Feature World
Describes the way in which OpenGIS Features are built up from geometry, attributes and reference systems.

Feature Collection World
The way in which OpenGIS Feature Collections are composed of OpenGIS Features. Whole of Feature Instances, Feature Schema and Project Schema.

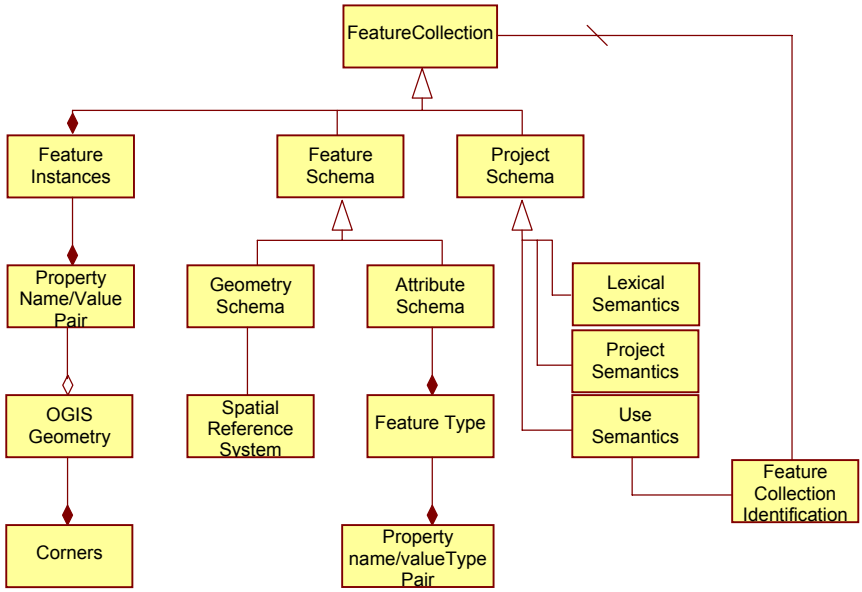


figure 5 abstract feature model (opengis) [15]

3.3.2 Complex Features

Complex Features are features composed of other features, of which the spatial values of the feature attributes together form a complex value, or between which topologic relations are defined. Complex features consist of *feature primitives*, to be subdivided in feature nodes, feature edges and feature surfaces, each with its own feature attributes.

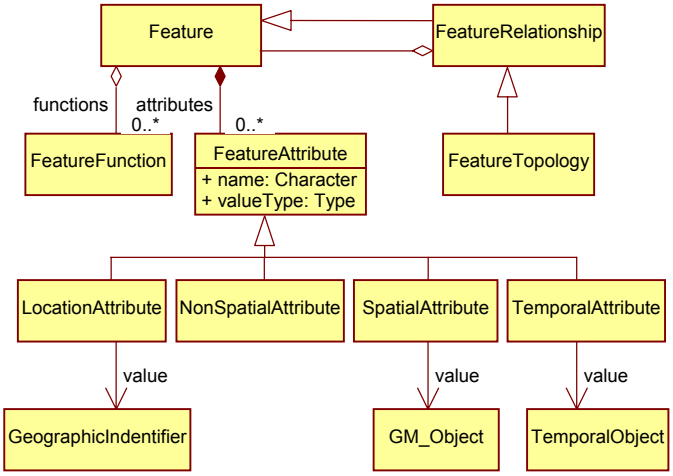


figure 6 general feature model (iso /tc211) [15]

3.4 Topologic models

Topologic models are based on the graph theory. Basic elements of the graph are *nodes* and *edges*. From these basic elements the next elements are to be distinguished [8]:

- *polyline*: a continuous chain of edges
- *graph segment*: a polyline of which in the intermediate nodes at most two edges come together
- *polygon*: a closed polyline (or a combination of closed polylines)
- *area segment*: the space enclosed by a polygon
- *face*: an area segment that is not intersected by other polygons

Based on this a topologic model for geometric objects can be set up.

Point objects are represented by nodes, line objects by edges and/or segments and area objects by faces. Nodes form the beginning and end of edges or represent point objects or do both. Edges or graph segments represent line objects or form the boundary between area objects or do both.

It is necessary to manage a few conventions, for example:

- All points are represented by nodes and every node represents at the most one point object.
- All lines are represented by edges or segments. Their starting and end node have to differ. Each edge/segment is part of at most one line object.
- At most there is one edge for every pair of nodes.
- Edges/segment do not intersect.
- Point objects that are represented by a detached node (meaning from where not any edges are coming) are situated in an area object.

3.4.1 Abstract Models

A (formal) vector data model for a topologic structure is the *Single-valued vector maps* [8].

In this model, shown in figure 7, point objects are represented by nodes, area objects consist of one or more faces and line objects consist of one or more segments. Further crossing lines are taken into account by modelling a crossing point with references to the two concerning curves. Detached nodes are situated in a face. This finds expression in the ISIN relation.

This model can be simplified by replacing the segments with the edges of which they consist and also by leaving out the faces. All relations with the faces are made directly with the area objects (figure 8). The consequence is that the edges bear practically all topologic relations. Now the ISIN relation has to be made between the point object and the area object.

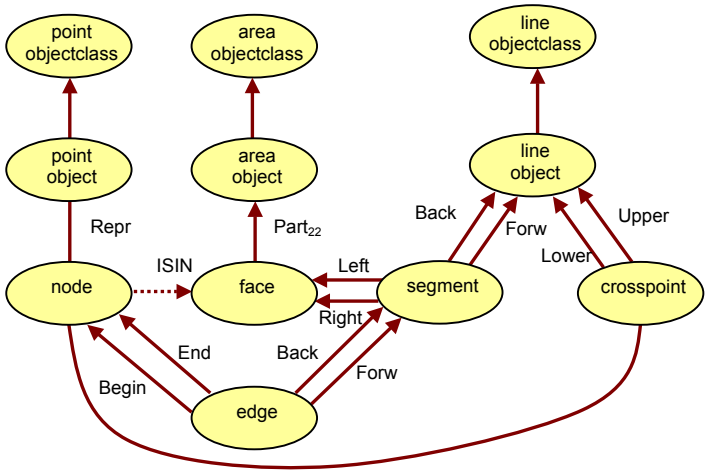


figure 7 general Single-valued vector maps [8]
The arrow indicates an n on 1 relation.

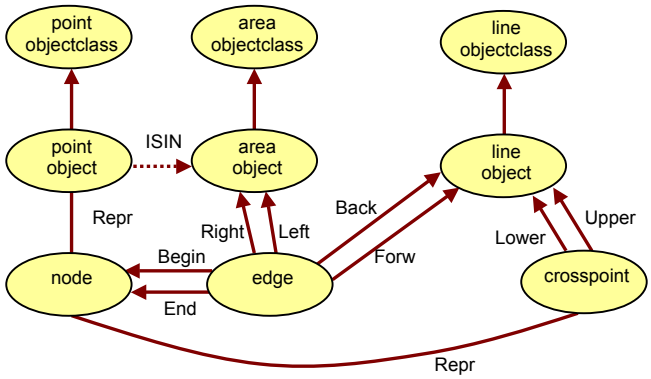


figure 8 simplified general Single-valued vector maps [8]
The segment object (from figure 7) is removed.

3.4.2 Implementation models

Topologic data structures consist of a list of nodes, a list of edges and a list of faces. A choice has to be made which list will be taken as starting point for the topologic relations.

With due observance of that, the next classification is to be made [1]:

- *Face topology*

For every face the overview of faces contains a (ordered) list of the edges that borders the face.

- *Node topology*

For every node the overview of nodes contains a (ordered) list of edges that meet in the node.

- *Edge topology*

For every edge the overview of edges contains the nodes that border the edge and the faces of which the edge is found in the boundary. (in 2D 2nodes and 2 faces belong with every edge.)

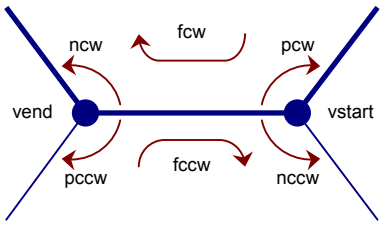
Well-known topology implementation models are [2][9][11]:

The *wheel topology* describes any face by an ordered list of the edges of which it consists (face topology).

The *left-right topology* (an edge topology) has a reference for every edge to the left area and a reference to the right area.

The *winged edge topology* or *chain topology* is an edge topology in which the edges also contain references to the next (and previous) edge of the faces the edge is part of, beside the reference to the adjacent nodes and faces. The list of faces has to contain a reference to a first edge for the outer ring (and references to a first edge for every inner ring) for every face, so that the remaining edges can be passed through. The direction of the edge in a face is retrieved by comparing the references to the faces or nodes. When the list of nodes also contains a first edge, then all edges that meet in that node can be found by alternately passing through the next and the previous edge.

When an edge is divided in two half oriented edges, then *the half-edge topology* is created. Every half-edge also contains, next to the references to the face it is part of, and to the previous and next edge, a reference to the other half.



vstart	start vertex, begin node
vend	end vertex, endnode
fcw/	face clockwise, right face
fccw/faw	face counter (/anti) clockwise, left face
ncw	next (edge) clockwise, next right
nccw/naw	next (edge) counter(/anti) clockwise, previous left
pcw	previous (edge) clockwise, previous right
pccw/paw	previous (edge) counter (/anti) clockwise, next left

figure 9 winged-edge topology [3]
Each edge contains references to the 4 adjacent edges with which it shares a node and a face. This is sufficient to reconstruct a face, or to retrieve the successive edges of a face.

3.5 XML

A short description of the XML [18] developments comes next. In appendix A.1 a description of XML of the specification in terms of terminology and syntax is included.

3.5.1 What is XML

Extensible Markup Language (XML) was developed in 1996 by the XML Working Group under supervision of the World Wide Web Consortium (W3C). It is derived from *the Standard Generalized Markup Language* (SMGL) and is meant for use on the internet and as successor of the HTML. Because SMGL is very extent, one of its subsets is contemplated. The result is XML. Therefore XML is compatible with SMGL.

A *Markup Language* is a formal mechanism to structure data in *elements* in a tree structure (see figure 11) by means of *markup*, among which *tags*. Tags border elements (see figure 10). XML is a standard specification to realize that. The difference between XML and HTML is that HTML has a fixed set of tags and that the semantics of the tags is also fixed. That brings along a few disadvantages of which the most important one is that bringing out a new version of HTML with a new required set of tags and semantics is a slow process and always lies behind on the demand of browser developers and users.

XML solves these problems by specify neither semantics nor a tag set. On the contrary XML has a mechanism to define tags and their grammatical structure. This is possible by defining *Document Type Definitions* (DTDs) [18] or alternative schemas (like *XML Schema Definitions* [20]) and including them in XML documents. That makes XML extendible and because of that XML is a meta-language for markup languages. With assistance of XML arbitrary data structures can thus be described.

With the development of XML especially simplicity is pursued:

- XML has to be transported and read over the internet just as easily as HTML.
- XML has to be usable in a breed veriety of (internet-) applications.
- XML software has to be simple to develop. For this reason XML must be EBNF conformant. Because of that XML is simple to parse.
- XML documents have to be human readable and understandable
- XML documents have to be simple to create.

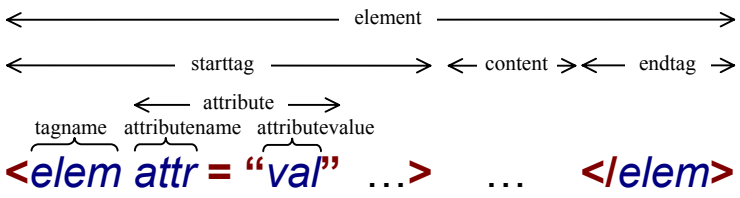


figure 10 XML elements
Elements are bounded by a starting tag and an end tag. Elements can contain attributes.

3.5.2 XML family

After bringing out XML 1.0 in 1998, various Working Groups of the W3C have been engaged with extensions of XML of general use. Amongst others:

XML Namespaces
When XML documents are composed of elements coming from different DTDs, name conflicts can occur. XML Namespaces should solve this by rendering universal names to elements.

Each used namespace has to be declared by joining a prefix and a URI together. This is possible in a arbitrary element by means of attributes **xmlns:prefix="URI"**. Prefixes can be chosen freely. Within the element the tags of DTDs can be used with **<prefix:element>** (and **</prefix:element>** and **<prefix:element/>**).

In essence XML solves possible name conflicts by putting a "**:**" in front of the tag names of a prefix.

XML Query
Query facilities to retrieve data from XML documents or XML databases.

XML Schema
XML Schema [20] is an alternative for DTD for tying XML documents to a grammar that works even more specific than DTD, by specifying data types for attributes and content for example. (see appendix A.2)

XML Linking
These are powerful hypertext link possibilities, for example external links, in which many to many links are possible [19]. (see appendix A.3)

DOM
The *Document Object Model* is a platform and language independent interface for programs and scripts and serves to parse/manipulate XML documents. XML documents then are to be approached as objects in a hierarchic tree structure. In this way every XML element is a node in the tree with a parent-node and child-nodes. Therefore this approach is named *tree-based* (see figure 11).

An alternative interface is the *Simple API for XML* (SAX). This is not a product of the W3C. SAX reads through the XML document and generates *events* for every XML tag or other fragment. The user needs to implement what should happen at a certain event. If necessary the user himself has to make an object structure on the basis of the events.

RDF
Resource Description Framework is a vocabulary (XML syntax) to describe varying (web-)metadata by rendering properties to resources (this can be anything). Relations between resources are modelled as *classes* and *properties* with *RDF Schema* (RDFS). The difference between RDFS on the one hand and DTDs XML schemas on the other hand is that RDFS defines an (object) data model, and a DTD or XML schema defines a syntax.

CSS
Cascading Style Sheets is a mechanism to define lay-out styles for web documents (HTML and XML). A CSS contains rules (*CSS rules*). Every rule assigns *values* to *properties* of certain HTML/XML elements (*selectors*). In this way lay-out and content can be separated.

XSL
Extensible Stylesheet Language also serves to define stylesheets, but is, in contrary to CSS an XML vocabulary. Moreover, it has more powerful possibilities at its disposal. It consists of two parts:

- *XSL Transformations (XSLT)* to transform XML documents from one vocabulary to another.
- *XSL Formatting Objects* to present XML documents on a medium.

XHTML
Extensible HyperText Markup Language is a redefinition of HTML 4 in XML. So the documents have to be well-formed (see appendix A), in contrary to HTML. XHTML 1.0 defines three DTDs:

- *XHTML Strict* (contains no lay-out mechanism)
- *XHTML Transitional* (contains lay-out mechanisms)
- *XHTML Frameset* (for the use of frames)

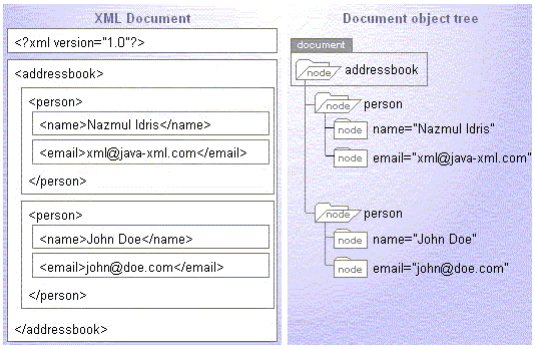


figure 11 DOM tree
The hierarchic tree structure of elements

4 Current situation: GML 2.0

This chapter provides an overview of the current GML specification (version: 2.0, status: OGC Implementation Specification) [13]. First the structure of GML comes up for discussion, and then descriptions of the different parts follow.

4.1 Structure

The intended aim of GML 2.0 is the creation of an open platform to define feature schemas and objects. The specification is based on the Simple Feature Model of OpenGIS. The consequence is that GML is not able to model topology. Now GML does permit complex non-geometric properties and associations between features, but as for the geometric attributes only simple geometry is possible.

The GML specification defines its vocabulary as XML Schemas. GML defines three XSDs:

- Geometry.xsd (geometric types and elements)
- Feature.xsd (feature types, including feature collections)
- Xlink.xsd (xlink attributes)

The XSD feature.xsd depends on geometry.xsd and geometry.xsd depends on xlink.xsd. Xlink.xsd is defined by the OGC, conform the W3C XLink specification [19], with the annotation that it can/will expire when W3C itself sets up a schema for XLink.

On the basis of these schemas application-specific schemas must be defined, by defining types derived from the GML feature and geometric types, and/or by declaring elements.

- The GML specification uses UML to visualize schemas. The relation between UML and GML is unambiguous here:
- A GML feature or geometry type corresponds with a UML class (UML class = GML type)
- A GML feature or geometry element corresponds with a UML class (UML instance = GML element)
- A GML feature property corresponds with a UML attribute (UML attribute = GML property) or with a UML association, in particular an aggregation (UML role = GML property)
- A UML generalization corresponds with a subtype derived from a GML type. Generalizations of the stereotype <<restriction>> are derived in GML by means of restriction.

4.2 Geometry XSD

In geometry.xsd GML specifies eight geometric elements and types for encoding instances of geometries, all derived from the abstract type **AbstractGeometry**. These are described in table 3.

Element	
Point	a point
Linestring	a curve, build of straight curves
Box	a rectangle encoded by means of 2 cornerpoints
LinearRing	a closed curve
Polygon	a polygon possibly with holes
MultiGeometry	a collection of geometric elements
MultiPolygon	a collection of polygons
MultiLineString	a collection of curves
MultiPoint	a collection of points

table 3 GML geometric elements

In figure 12 the graphical representation of the model is shown.

The elements **Point**, **Box**, **LineString** and **LinearRing** have the sub-elements **coord**, that in their turn have the sub-elements **X**, **Y** and **Z** for individual coordinates. The alternative is the element **coordinates** with one string for the coordinates.

The element **Polygon** contains one sub-element **outerBoundaryIs** for the boundary and for each hole a sub-element **innerBoundaryIs**. Both contain a **LinearRing** element.

The element **MultiGeometry** (instance of the type **GeometryCollection**) contains a set of geometries, both encoded by the sub-element **geometryMember**, that contains in its turn a geometric element. The elements **MultiPolygon**, **MultiLineString** and **MultiPoint** are sets just like **MultiGeometry**, however they can contain merely one type of geometry.

4.3 Feature XSD

The schema feature.xsd defines the abstract elements **AbstractFeature** and **AbstractFeatureCollection** and corresponding types. The corresponding feature model is shown in figure 13. For the geometric feature properties there are predefined property-elements: a general **geometryProperty** element and a more specific element for any type of geometry (formal).

Additionally the schema declares a number of property elements as synonyms of the general property elements (descriptive), that more or less describes the use of the property. (Table 4)

Formal name	Descriptive name	Geometry element
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 4 GML property elements
A property element contains a corresponding geometry element. A (formal) property element and several descriptive elements belong with each geometry element.

The **FeatureCollection** type is a subtype of the **Feature** type, but it contains **featureMember** elements.

For relations between features there is the **FeatureAssociation** type. The **featureMember** element is an instance of this type. Elements of this type can contain a feature or can refer to a feature. In order to make references possible, this type contains the *simple link* attributes of Xlink (**href**). Besides, this type contains the attribute **remoteSchema**, with which the URI of the schema fragment to which the element to be referred has to validate to, is fixed.

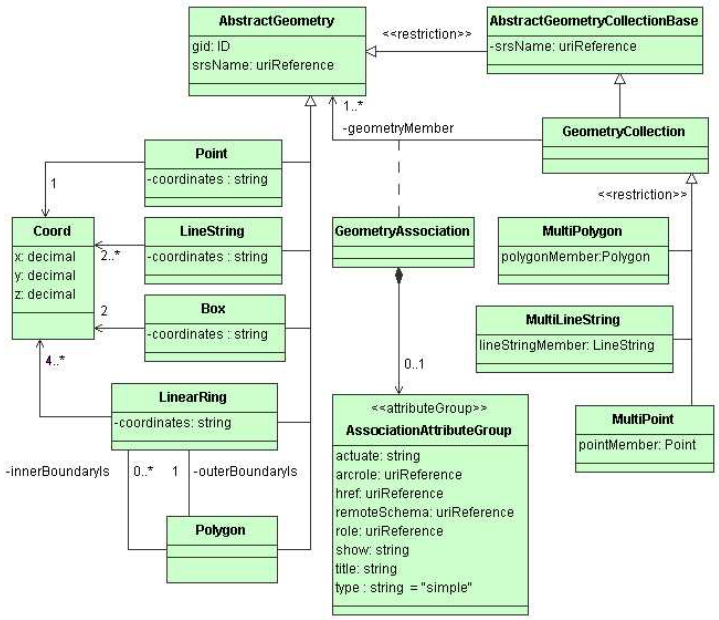


figure 12 GML geometric model
The GeometryAssociation allows references to a Geometry for geometries within a GeometryCollection.

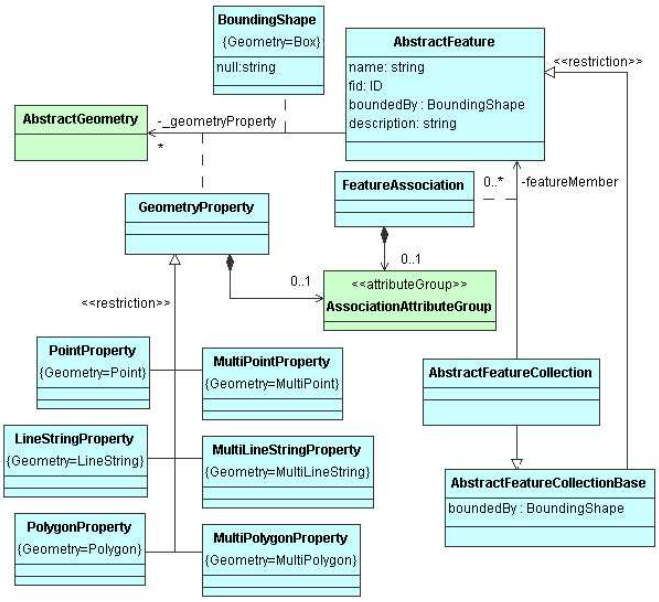
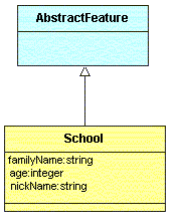


figure 13 GML feature model
A feature can consist of one or more GeometryProperties. FeatureCollections consist of several Features.

4.4 Application specific XSDs

The schema feature.xsd only defines types and declares abstract elements. In order to encode GML features an application specific XSD will have to be specified (figure 14). Own-written XSDs have to import the GML namespace and declare a *targetnamespace* for themselves.

The feature types that are defined in this, must be derived from one of the GML types: **AbstractFeatureType**, **AbstractFeatureCollectionType** or **FeatureAssociationType**. The feature properties are allowed to be simple or complex. For the geometric feature properties one can refer to the predefined property elements, but also self-defined elements of one of the geometric types are allowed. New geometric property types must be derived direct or indirect as a subtype of **GeometryPropertyType**. Also own-written geometric types are definable, provided, they are derived direct or indirect from **AbstractGeometryType** or **GeometryCollectionType**. Moreover the attribute group **AssociationAttributeGroup** has to be included in it.



XSD:

```
<element name="Dean" type="ex:DeanType" substitutionGroup="gml:_Feature"/>
<complexType name="DeanType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element name="familyName" type="string"/>
        <element name="age" type="integer"/>
        <element name="nickName" type="string" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="gml:location"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

GML:

```
<Dean fid="D1123">
  <familyName>Smith</familyName>
  <age>42</age>
  <nickName>Smithy</nickName>
  <nickName>Bonehead</nickName>
  <gml:location>
    <gml:Point>
      <gml:coord><gml:X>1.0</gml:X><gml:Y>1.0</gml:Y></gml:coord>
    </gml:Point>
  </gml:location>
</Dean>
```

figure 14 example XSD en GML fragment
The XSD defines a new (sub)type 'Deantype' as extension of 'AbstractFeatureType'. Afterwards an element 'Dean' of this type is declared. The XML fragment 'Dean' contains the newly defined elements next to the elements and attributes that are part of the _Feature element.

5 Required situation

This chapter describes the requirements the GML specification has to meet, and presents (basic) models to achieve this.

With GML 2.0 there is a possibility to encode complex objects as a GeometryCollection for example, in which the **geometryMember** elements contain the primitives. However by integrating a topologic model in GML, it is possible to encode complex objects in an efficient way, because in such a model, coinciding coordinates are included only once. Moreover, topologic relations between primitives are made explicit.

By using the winged-edge model as a topologic model, also the number of necessary references between the primitives is kept minimal. The geometric model of GML will have to be adjusted for this.

5.1 Geometry

In the GML class model all geometric primitives are modelled as a sub-class of AbstractGeometry, without further division. However implementation of topology requires a distinction between primitives of different dimensions. The geometric model must thus be extended with at least these (abstract) classes, as shown by figure 15. The original geometric classes are modelled as sub-classes of the introduced (abstract) classes, which makes that extension has no influence on existing GML documents.

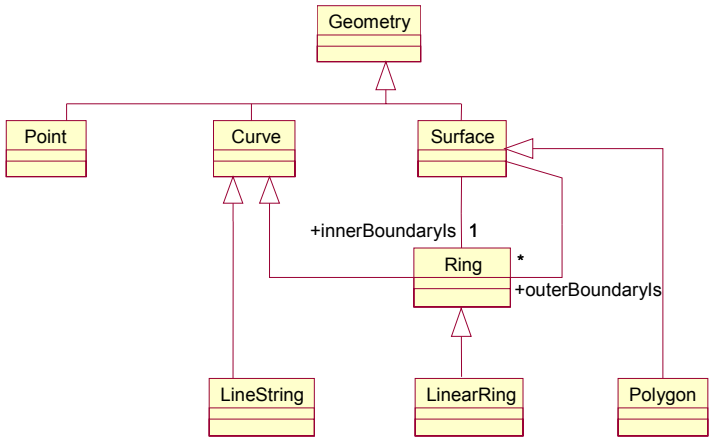


figure 15 geometry
Distinction between the three dimensions has become necessary.

5.2 Topology

Topology is modelled analogous to geometry, by distinguishing topologic primitives. The topologic relations lie in the association between these primitives. The topologic model must represent this. The winged-edge model is taken as starting point. The Xlink specification lends itself to encode networks of *edges* as *extended links*, in which the Xlink *arcs* are used for the *edges*. Here Xlink is not chosen because of the number of relations of individual primitives., Although the Xlink attributes could be supplemented with attributes for the remaining relations, it is chosen to leave out Xlink here.

In figure 16 the winged-edge model of figure 9 returns. However, the relations pcw and paw are left out. The reason for this is that the relations naw and paw are sufficient to reconstruct the faces. Moreover pcw and paw are easily derived from the remaining relations. (An edge is the pcw or paw of its ncw or naw.)

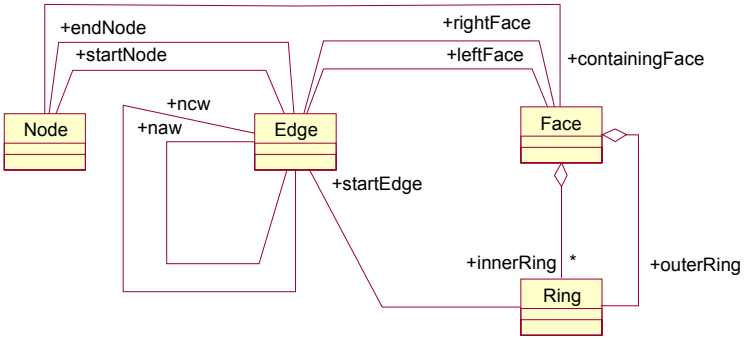


figure 16 Chain topology
A UML translation of the winged-edge model (figure 9).

6 Alternatives

In the previous chapter basic models were supplied for geometry and topology. For the new GML model that will be based on these basic models, the relation between geometry and topology still has to be fixed and an appropriate feature model has to be set up. This is possible in a number of ways. A few alternatives are worked out below.

6.1 Complex

The model for complexes has to combine geometry and topology. There are numerous possibilities for modelling this. Below a number of options are contemplated here. Each introduces a class *Complex* as aggregate of primitives. In this a *Curve*, as part of a complex, does not need to contain coordinates. In case of missing of coordinates there is a straight line of which the starting point and endpoint can be retrieved.

6.1.1 option A: topology contains geometry

In this option each topologic primitive contains a geometric primitive of the same dimension, that is modelled here as aggregate.

Advantage:

- It is possible to describe topology without geometry.

Disadvantage:

- Topology stands above geometry.

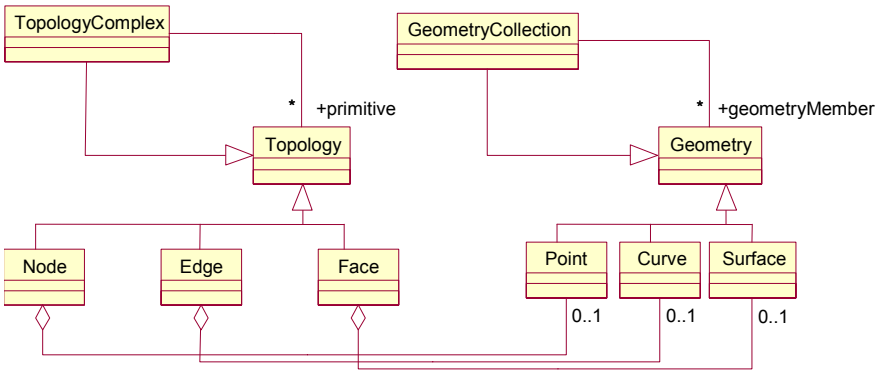


figure 17 option A: topology contains geometry

6.1.2 option B: topology as an association class

In this option the class *Topology* is an association class. *GeometryComplex* can possibly be modelled as sub-class of *GeometryCollection*. In order to guarantee that topologic primitives and the geometric primitives they associate, are from the same dimension, supplementary restrictions still have to be included in this model.

Advantage:

- Topology does not stand above geometry

Disadvantage:

- The description of topology without geometry is not possible, unless the geometric classes realize that.

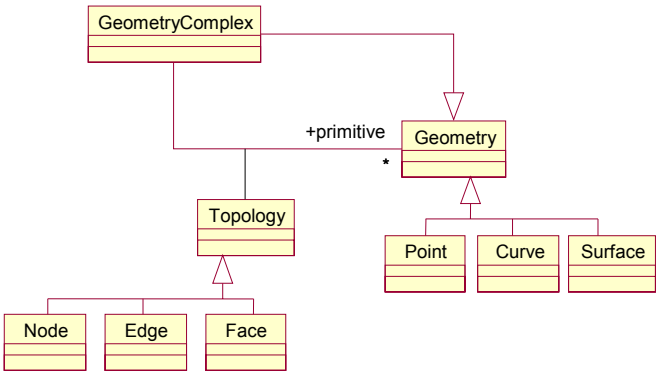


figure 18 option B: topology as an association class

6.1.3 option C: topology within geometry

In this option topology is interwoven in geometry. Each geometric primitive class contains a group of topologic attributes.

Advantages:

- Integrated geometry and topology makes references between both unnecessary.
- There is no distinction between geometries that are part of a compex and other geometries, making this transparent for features.

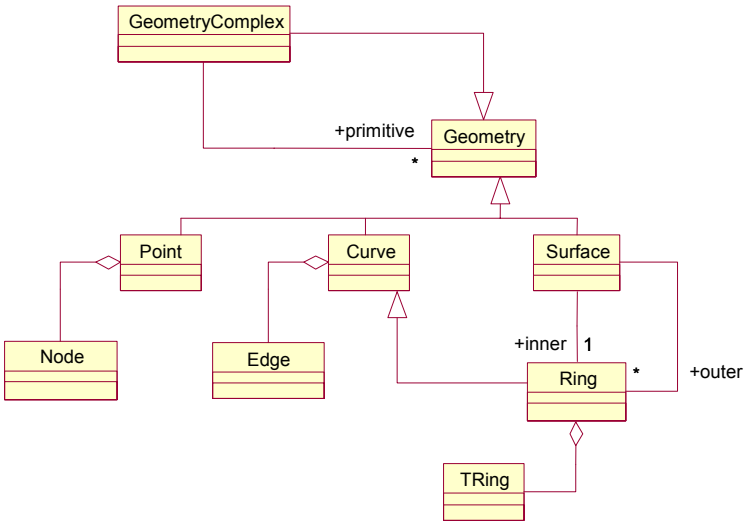


figure 19 option C: topology within geometry

6.1.4 option D: topology separate from geometry

In this option geometry and topology are modelled in principle totally separate from of each other as *TopologyComplex* and *GeometryComplex*. Each topologic primitive contains (if necessary) a reference to the corresponding geometric primitive and vice versa.

Advantage:

- Geometry and topology are completely separated, as in the abstract OpenGis specifications

Disadvantage:

- Measure of redundancy

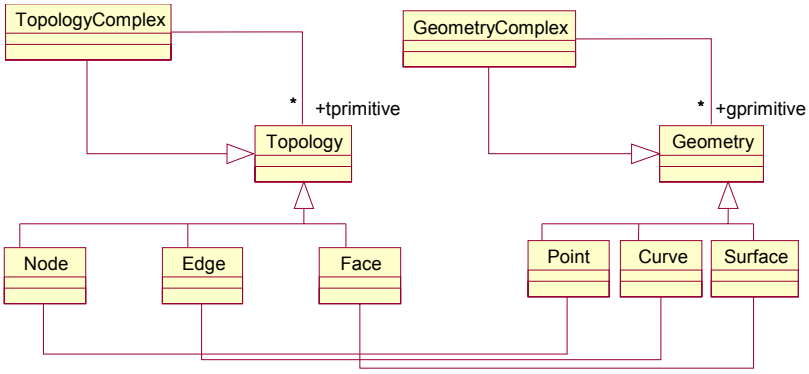


figure 20 option D: topology separate from geometry

6.2 Feature

With one of the represented models in the preceding, geometry and topology can be represented. Just as important is how all this can be used in accordance with features. In the current model the relation between *Geometry* and *Feature* lies in the association class *GeometryProperty* (see figure 13). Below a few options are shown. In this, sets of features of which the geometries form a complex are nothing else but *FeatureCollections*.

6.2.1 option I: topology on the feature level

Option I introduces a *FeatureComplex* consisting of several Features, each classified as *FNode*, *FEdge* or *FFace*. These take over the role of topologic primitives. Thus this option is not combined with one of the complex options.

Advantage:

- The relation between a feature and its topologic adjacent features is visible.

Disadvantage:

- Surface geometries must be reconstructed on feature level
- A feature can only be part of one complex.
- Empty features for primitives that are not part of a feature

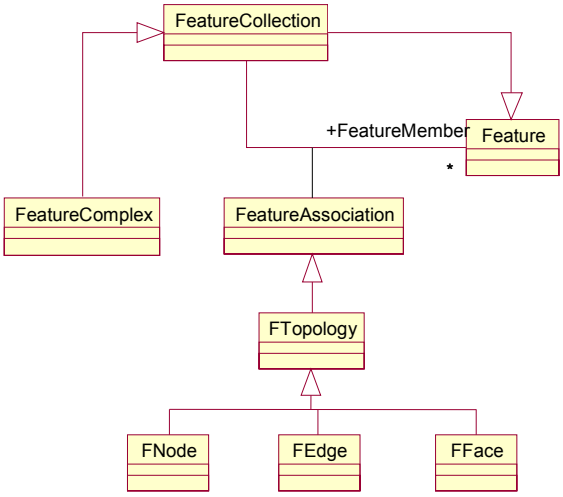


figure 21 option I: topology on the feature level

6.2.2 option II: properties for topologic primitives

Here *Features* can also contain *TopologyProperties*, beside or instead of *GeometryProperties*, that in their turn contain a topologic element or refer to it.

Advantage:

- There is a direct reference to a Topology instance

Disadvantage:

- A distinction is made between geometries that are part of a complex and other geometries.

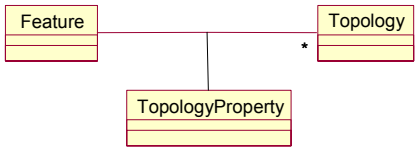


figure 22 option II: properties for topologic primitives

6.2.3 option III: property for complexes

Here *Features* can also contain *ComplexProperties*, that contain can entire complex. ComplexProperty can be included as a sub-class of GeometryProperty.

Advantage:

- The association between Feature and Geometry is maintained as in the current GML model

Disadvantage:

- There is no reference to a Topology instance, so that the Topology instance has to be retrieved when reconstructing the Surface geometries. The complex model has to make this possible.

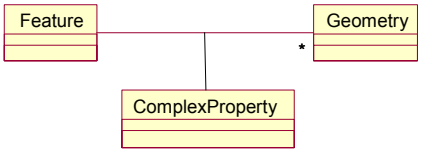


figure 23 option III: property for complexes

6.2.4 option IV: primitives contain features

Each topologic primitive contains or refers to a Feature (possibly several Features)

Advantage:

- A geometry/topology can retrieve the features to which it belongs directly.

Disadvantage:

- This is a huge adjustment of the current feature model

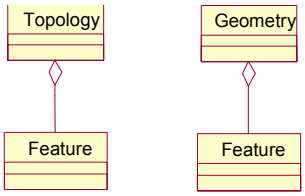


figure 24 6.2.4 option IV: primitives contain features

6.2.5 option V: former situation

The feature model is not supplemented as for *GeometryProperties*

Advantage

- The current feature model is held

Disadvantage

- Features are not able to contain complexes as its properties

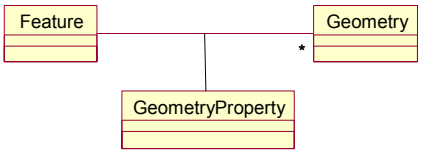


figure 25 option V: former situation

6.3 Location complex

Beside abovementioned options the way in which complexes can be encoded in relation to features also has to be contemplated. There seem to be three possibilities, namely:

1. outside features, as one complex whole
2. inside features, as one complex whole
3. inside features, primitives spread over features

Possibility 3. has as advantage above the others that with each primitive the corresponding feature can be found. In this manner each feature can trace all adjacent features. A disadvantage is that a solution has to be sought for the inclusion of primitives to which no Feature belongs, for example by means of ‘empty’ features.

In the possibilities 1. and 2. features must refer to the geometries/topologies that are part of a complex. The disadvantage is that geometry can never be included with the feature.

7 Recommended choice

The choice of chain topology will contribute a lot to the aim of encoding complexes efficiently. The mutual differences between the submitted alternatives will probably contribute a lot less. Nevertheless compactness stays an important criterion in order not to annul the efficacy chain topology brings along (criterion 1). Also the criterion redundancy keeps playing a role. The model should allow as few inconsistencies as possible (criterion 2). The feature model is not allowed to make distinctions between geometries that are coincidentally part of complexes and others (criterion 3). Finally possible artificial constructions, such as empty features, should be avoided (criterion 4).

Based on these criteria *option C: topology within geometry* is preferable to A, B, and D. It is more difficult for option A and D to give transparency between simple geometries and geometries that are part of complexes. Option D leaves to much room for inconsistencies between geometry and topology. Option C appeals the most, because of its compactness. Topology is an aspect of geometry and is modelled entirely integrated.

Location possibility 1 or 2 is preferable because of the disadvantage of possibility 3. A slight preference goes out to *possibility 2. inside features as one complex whole*, because it allows features, of which the references reside within the features. This choice does not rule out possibility 3. for that matter.

As for the feature model, *option III: property for complexes* links up well with the combination option C and possibility 2. Further it affects the current model minimally, which makes that this model links up well with the basic assumptions of the current GML 2.0 specification. In this way a Feature is able to contain a *complexProperty*. Other features can contain geometric properties that refer to geometries that possibly are part of a complex. This makes no difference with this option.

In order to make possible that features retrieve their adjacent features anyway, the topologic primitives can be extended with a reference to the feature to which they belong.

8 Further elaboration

In this chapter the chosen complex and feature model from the previous chapter are worked out more profoundly, to a complete model. The new complex model offers a number of new possibilities, to which are given shape in the next paragraphs.

8.1 Composite

A complex can form one line or area. The geometry then is a *Curve* or *Surface* as well as a *Complex*. In the Abstract Specification [14] this was modelled as *Composite*s, sub-classes of both a primitive and a complex class. An obvious solution is the introduction of the classes *CompositeCurves* and *CompositeSurface* that contain a *Complex*.

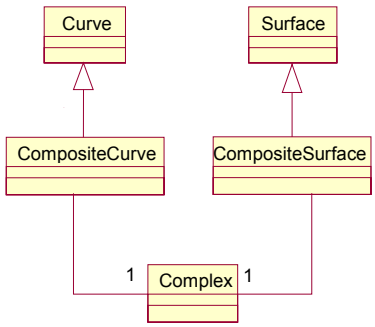


figure 26 composite
A Composite is a Curve or Surface that is described as a Complex.

It is imaginable that within complexes a few primitives together form one geometric curve or surface together. This can be encoded with the GeometryCollection or one of its subclasses, but from this it does not appear that it is about one curve or surface. These geometries are in essence also composites, though with references to the primitives. As the complex already exists elsewhere, it must be sufficient to include only primitives of the highest dimension.

8.2 Complexes hierarchy

A primitive in a complex can also be a complex itself. Especially with a *Surface* a sub-division in sub-surfaces can be desirable: a sub-complex. Here the *composite* classes can be a solution, for these can contain the sub-complex. A supplementary solution is to provide the primitive with a reference to its sub-complex, or reversely the (sub-)complex with a reference to its super-complex.

8.3 Complex features

Beside the until now contemplated topology, also relations between features without the geometry can be desired. Especially networks such as roads and waterways can be encoded with this. Just like an edge connects two nodes and separates two faces, a feature can form a connection or in contrary a separation between two other features. Examples are a river between a lake and a sea, a ditch between two pastures and traffic connections.

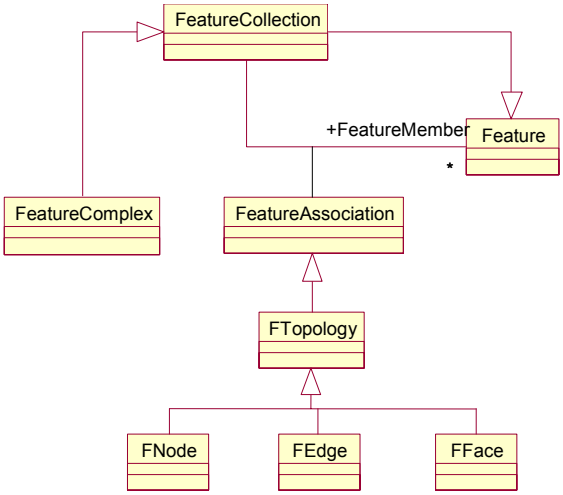


figure 27 complex feature
a FeatureComplex consists of a set FNodes, FEdges and FFaces.

8.4 Complete model

Fitting together the chosen complex and feature models, the above extensions and the current GML model, altogether produce the following:

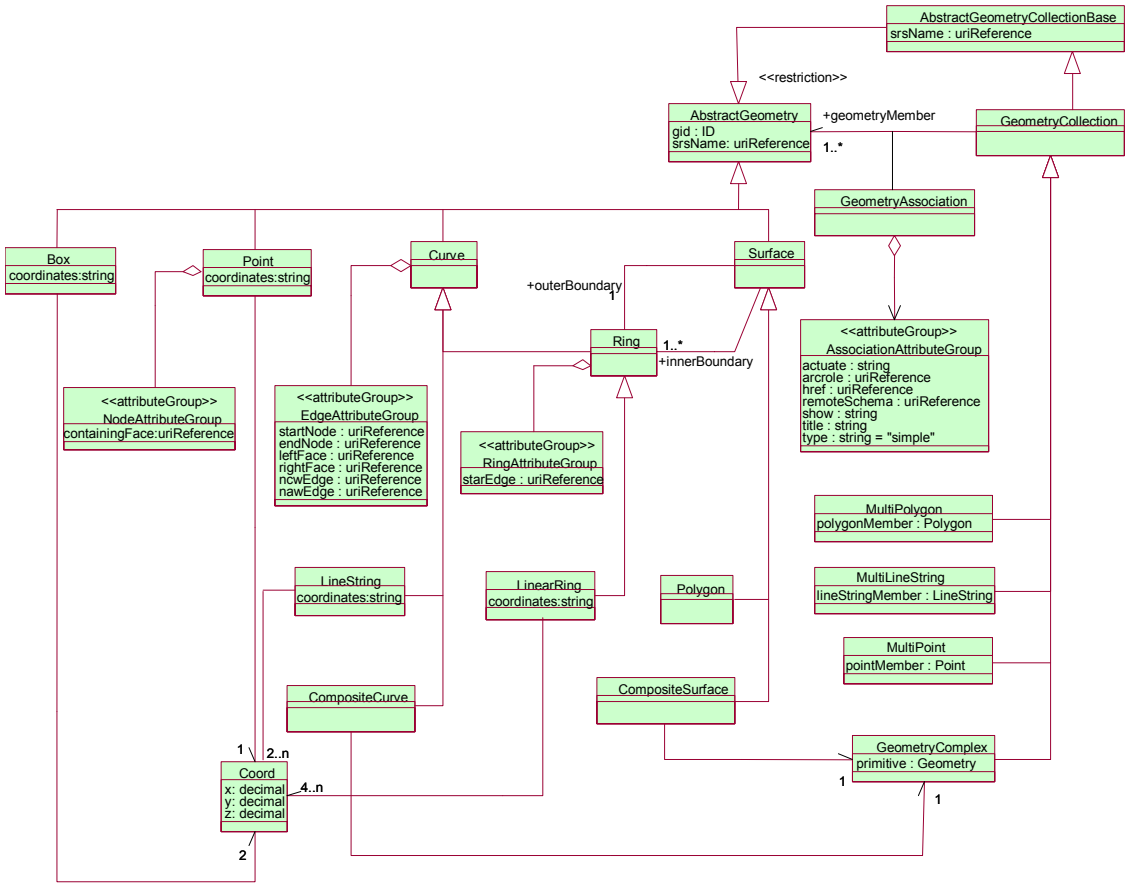


figure 28 new geometry model
Replaces the current geometry model (figure 12).

In the geometry model (figure 28) the greatest change concerns AbstractGeometry (and sub-classes). The classes Box and the sub-classes of GeometryCollection are maintained. GeometryCollection is supplemented with GeometryComplex.

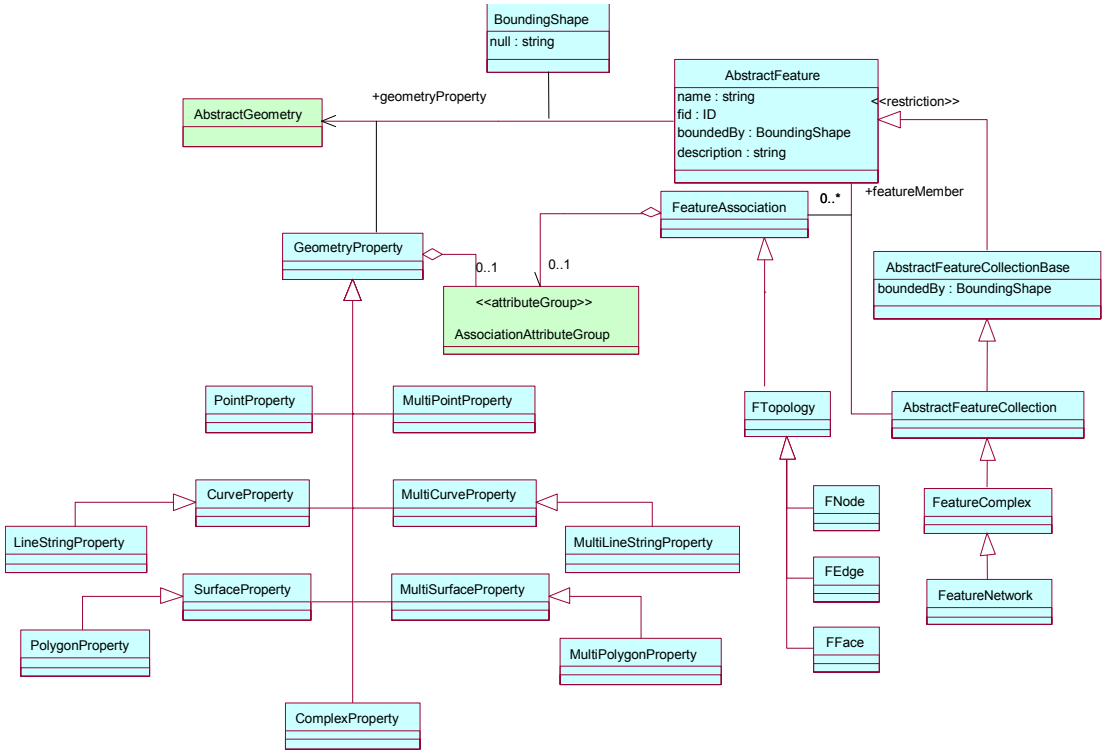


figure 29 new feature model
Replaces the current feature model (figure 13).

In the feature model (figure 29) the subclasses of GeometryProperty are supplemented. The greatest change here is the extension of FeatureAssociation with the subclass FTopology.

8.5 XSD schemas

On the basis of the new models the schemas geometry.xsd and feature.xsd are rewritten. As basis the original XSDs are taken. The design of GML 2.0 has remained intact.

At the conversion all classes in the UML model are defined as a XSD (complex) type. The class hierarchy of the UML models is translated by defining UML specializations as XSD types as a subtype (with extension or restriction).

XML elements of these types are declared and clustered in substitutionGroups for elements with the same super-type as much as possible.

The XSDs are written with the software package XML Spy [21], which offers a graphic interface for the development of XSDs. The XSDs are valid (thus well-formed), which is warranted by the package.

The result is included in appendix C.

9 Implementation

In order to prove the utility of the new GML model, next is proceeded to the use of GML to encode existing geographic data from a database. The implementation, as shown in figure 30, is divided in three parts: loading data, generate and visualise GML. These three parts are discussed in the sections 9.1 to 9.3.

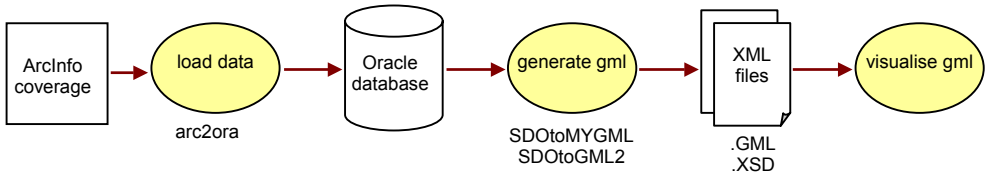


figure 30 architecture
The implementation consists of three part

9.1 Preparation: Load topologic data in database

As testdata a topologic (part of a) map of the Dutch Topographic Service (Topografische Dienst Nederland) is chosen. The testdataset was available in the ArcInfo coverage format of the GIS package ArcInfo [26] (version 8.0.2). Because of the size of the testdata, smaller testate are extracted from the original with help of ArcEdit (part of ArcInfo) (figure 31). In a fairly simple way ArcInfo can write out all spatial primitives in text format at will. The process of loading-in then also is divided into the parts: spit

1. write out geometric/topologic data from the testdata in ASCII (with ArcInfo)
2. load the ASCII format in predefined tables (with Oracle SQL Loader)
3. compute, calculate and include supplementary relations in the database

The testdata, and the above translations, comprise the most important topologic relations between the primitives. However a few relations that are relevant for the GML translation, namely the adjacent edges (ncw, naw, pcw, paw) are missing. It is chosen to calculate these relations in a third step in this stage of preparation.

Each of these parts is implemented as one or several scripts. The callings of these scripts are placed in a shell script (arc2ora.sh, see figure 32 and figure 33).

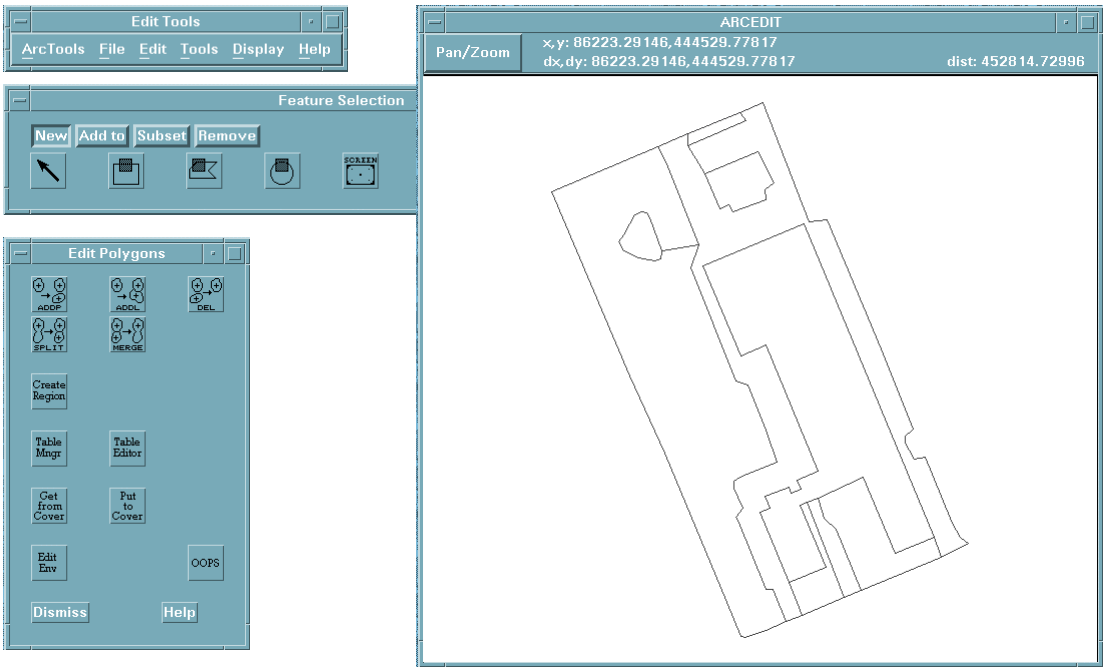


figure 31 screenshot ArcEdit
ArcEdit is used to create smaller datasets from the testdata.

9.1.1 Export testdata to ASCII

The testdataset forms a (planar) partition, in ArcInfo terms a coverage. ArcInfo offers a series of functionalities, in the form of commands for the arcshell, to approach and manipulate these. ArcInfo keeps up intern tables, amongst other for the topologic primitives. These can be selected, after which the records can be retrieved, and in this case the desired attributes can be written to a file as text. This must happen for nodes, edges and faces. In ArcInfo these are stored in (respectively) the tables with the extensions .aat (arc attribute table), .pat (polygon attribute table) and .nat (node attribute table). The attributes with topologic relations are most important, but the remaining attributes might also be interesting and are also exported for that reason. The geometry of the primitives as well turns out to be easy to export to in a text format of ArcInfo. The topology and geometry of the points demanded some extra work. The .nat table was absent, but could quickly be calculated by ArcInfo. In order to get the coordinates, a new coverage is made with only the points. The successive arc-commands to perform all this are placed in a parameterised shell script (unloadarc.sh), so that only one calling of this script with the name of the coverage results into six ASCII files (with the extensions .nodes, .edges, .points, .lines and .polygons).

9.1.2 Import ASCII data in database

The package Oracle Spatial 8i [27] contains the tool SQL Loader with this import function. This tool is called with a *control* file, that describes how and in which table the ASCII data have to be imported, and the ASCII datafile. In order to import the geometric and topologic data two perl scripts have been written (loadora_geom.pl and loadora_topo.pl). Both simultaneously process one of the ASCII-files of the previous step.

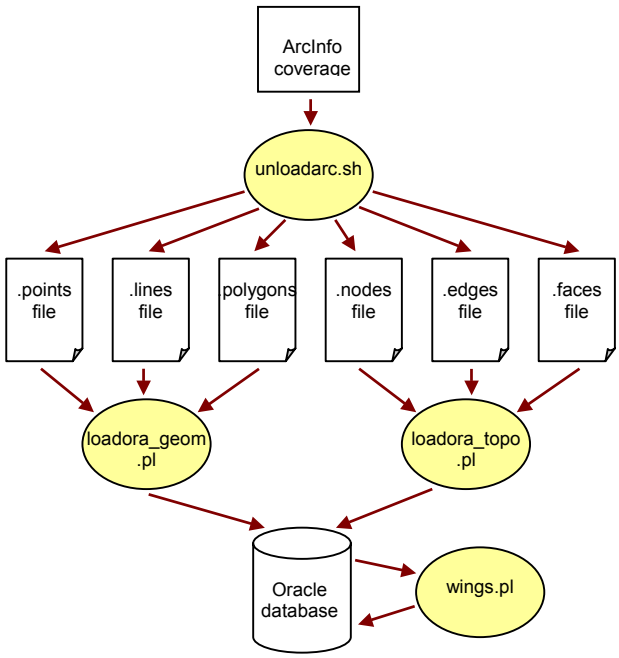


figure 32 arc2ora.sh script
Dataflow between the 4 (sub)scripts.

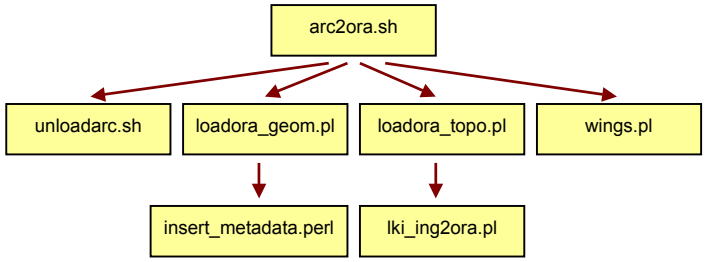


figure 33 scripts
Arrow indicates dependency (A→B, A calls B). Insert_metadata.perl and lki_ing2ora.pl are existing scripts.

Geometry

The geometries have to be imported by the script (loadora_geom.pl) as Oracle geometry object [27], of which the coordinates have to end up in variable arrays. To facilitate this, it is necessary to translate the text format of ArcInfo into a more suitable text format, of which the structure is simple enough to be

interpreted by the loader. During this process additional spatial characteristics are calculated and added to the new text data immediately, such as the *bounding box* and the *convex-hull* (not entirely correct: the outer edge is contemplated) for the polygons. Further the script makes data base tables, by means of SQL create table statements and the control file for the Oracle SQL Loader. After this the script can call the loader, that imports the data in the table that is made. Finally the script supplements the database with metadata about the bounding box and the precision of the data (with assistance of the script insert_metadata.sh).

Topology

The script for this (loadora_topo.pl) is simpler because the topologic data from step 1 have a simpler structure than the geometric data. It appeared that a script (lki_ing2ora.pl) that provided in the function to make control and data file (though for a conversion step of Ingres [25] to Oracle), already existed. By using this script only the table has to be made in Oracle and the loader to be called.

9.1.3 Calculation and supplementation of topologic relations

After step 1 and 2 the database contains 6 tables, of which 3 have topologic relations. The table with edges contains references to begin and end node and left and right surface. Compared to the extent GML model, a few relations are missing. For the edges these are the references to the adjacent sides (ncw and naw). Further the relation between begin side and the rings of a surface are missing. Also the nodes lying inside a surface are not provided of a reference. It is chosen to calculate this information and to add it to the database by means of SQL queries. These are placed in a perl script (wings.pl). The next strategy is applied.

General

- First a table *edge*; is created with only the sides that separate two different surfaces.
- Sides of which the two surfaces are equal, are placed in the table *bridgearc*.
- A table *node* is created from the table *edge* with the grade (number of connected edges) as attribute.

Calculate adjacent sides

- A table *ncwbase* is created with all combinations of 2 edges that meet the condition: a node and a face have to correspond in such a way that the second edge can be a ncw of the first. In a similar way the tables *nawbase*, *pwcbase* and *pawbase* are created.
- Double appearances in the base tables are filtered out by the extra condition: only one of the surfaces of both the edges are allowed to correspond, not both. This results in the tables *ncs*, *naw*, *pcw* and *paw*.
- Joining of these four tables in the table *wings*.

Calculate starting edges of outer and inner rings of surfaces

- Create the tables *parcbase* and *narcbase* (in a similar way as the adjacent sides) with for each side from *bridgearc* a reference to the next and previous edge it is joint with. As an extra attribute the face belonging to the previous/next edge is included.
- Filter out double appearances in these base tables by making a choice. This results in the tables *narc* and *parc*.
- Create the tables *innerring* and *outerring* by comparing the geometries of the surfaces which are associated by parc/narc. For this the spatial functions of Oracle [27] are used to define which of the polygons lies inside (of?) the other. The function SDO_RELATE is used on the convex-hull of the polygons that was imported in step 2 (see section 9.1.2).

Merge topology and geometry

- Merge the geometric and topologic primitives in the tables *_node*, *_edge*, *_face* and *_ring*.
- Place the non-topologic attributes in the table *feature*
- Remove all made helping tables

9.1.4 datamodel

The database that is now obtained after the calling of wings.pl contains the tables *_node*, *_edge*, *_face* and *_ring* and the table *_ffeature* (and *_effeature*) too. In figure 34 the coherence is shown.

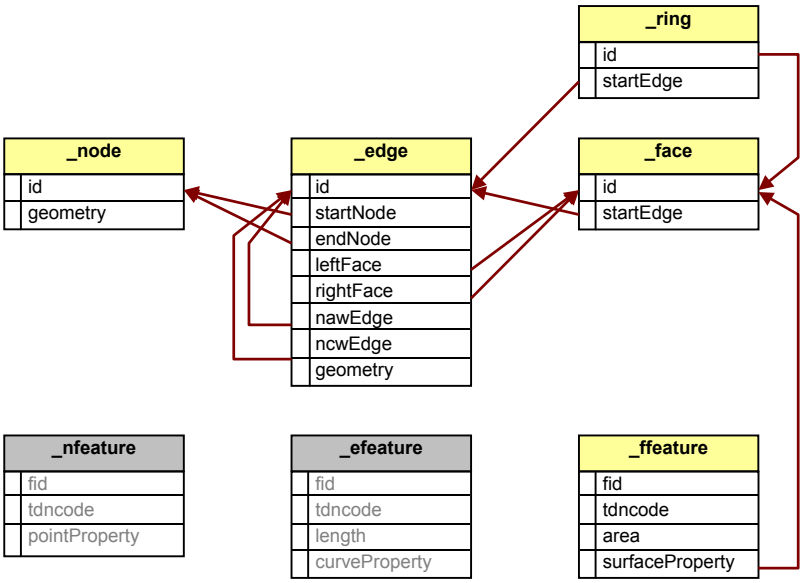


figure 34 datamodel of the testdata loaded in Oracle
_nfeature (for features with a point as geometry) and _efeature (for features with a line as geometry) are not worked out.

9.2 Generate GML from database

After the testdataset is loaded into the database and the necessary topologic relations are added, GML can be generated. For this a Java application (SDOtoMYGML.java) is written. This application globally does the next:

- Connecting to the database
- Retrieving the different tables
- Writing the XSD by using the attribute names and types
- Converting the attributes of the records of the different tables into the correct class one by one, and export to GML format.

To export to GML format, the class GMLGeometry.java is used. This can handle all geometric classes of the Oracle Spatial package. Successively *Point*, *LineString* and *Polygon* elements are coded within a *geometryComplex* element.

Because a surface is able to contain several rings, the application has to put them in order first. Then the feature elements are written out. Because in the preparing steps the tables are created into the desired form, with all supplementary topologic attributes, this application does not have to calculate any topologic relation itself.

For comparison a second Java application (SDOtoGML2.java) is written, which does the same, but maintains the GML 2.0 standard. It is to expected that the GML export of this application is larger in proportion than the first one.

9.3 visualize GML

In order to prove the correctness of the generated GML, it will have to be visualised. It is chosen to further extend a small existing Java GIS viewer (quickGIS) with GML functionalities. The basic aim was only to visualize GML. The extension consists of parsing a GML file and converting the acquired objects into the existing geometric objects of the viewer. For this task the XML packages of Xerces are used [23]. The parser was written for the extended GML in the first place, but is adjusted afterwards to also handle GML 2.0.

9.3.1 DOM

At first the DOM (Domain Object Model) approach is used, in which the DOM parser converts an arbitrary XML document to a DOM-tree. This tree than is recursively passed through to construct the desired objects. Because of the many references in the GML file it is necessary to pass through the tree several times. The first time all references between feature (-property) elements and geometries are searched. The second time all Point and LineString elements are converted in their equivalent objects of the application. After that all geometries are collected, in which the Polygon objects are reconstructed with help of the references.

9.3.2 SAX

After the DOM implementation also a SAX (Simple API for XML) parser is implemented. A SAX parser is event-based, so that an event-handler class (SaxGMLHandler.java) has to be written, which describes what needs to happen when an event occurs. The events that are implemented are *beginDocument*, *endDocument*, *beginElement*, *endElement* and *characters*. As a SAX does not make a tree-structure, the handler needs to build a structure itself. For this a package (franklin.gis) is written with classes for Java equivalents for the geometric and feature elements from the extent GML model. During the processing of the SAX events these objects are created, and joined together in the event *endDocument*.

Now it remains to convert these objects into the original object model of the viewer. The self-written package is set up in such a way that it can reconstruct the Polygons.

9.3.3 DOM vs. SAX

The DOM approach is simpler than the SAX approach as for parsing. The DOM parser returns a DOM tree with one function call, while the SAX parser has to build a structure itself. The advantage of the SAX approach however, is the simple manipulation of the obtained structure compared with the manipulation of the DOM tree.

Further the parser is refined to save all thematic properties (from the GML file) of the features with their geometries, and realize colouring according to a thematic property.

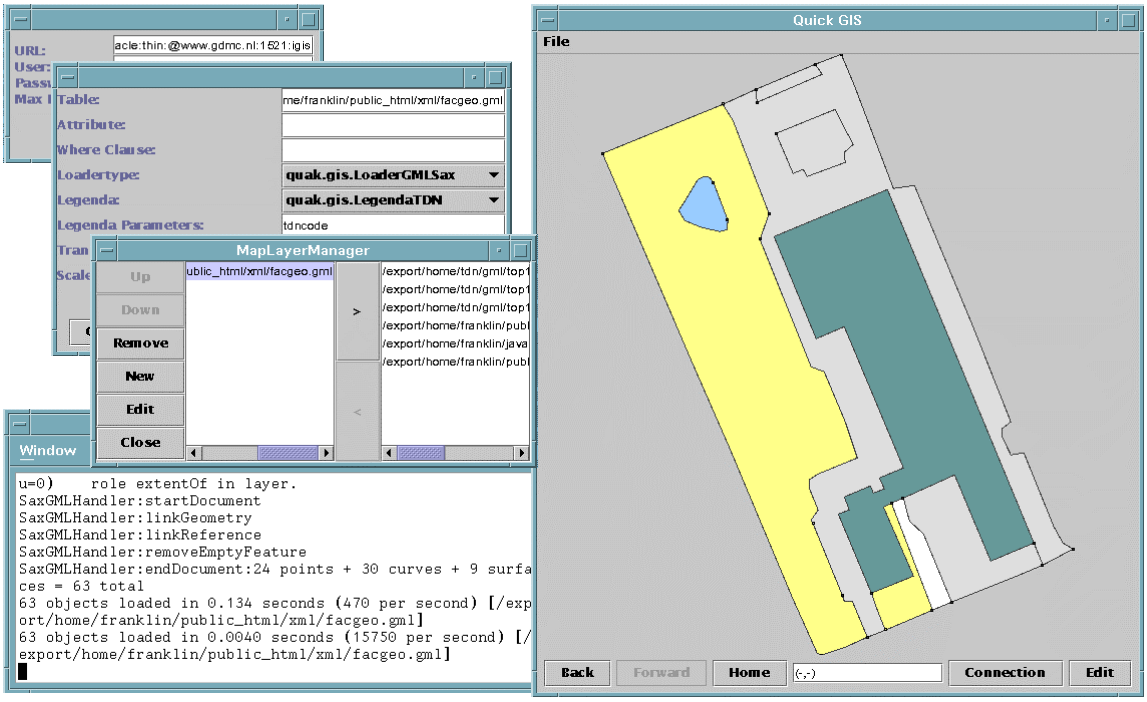


figure 35 QuickGis sreenshot
The application parses the stated gml-files and reconstructs the geometries to draw them. By means of the tdncode the geometries are coloured according to a legend of the application.

10 Conclusions, recommendations

The GML specification, developed by the OpenGIS consortium, offers an XML format to encode geographic data. In GML however specific mechanisms for encoding complex geometric objects (that is topology) are missing,. Therefore in this project GML is extended with topology, according to the winged-edge principle, which has resulted into a new GML model completely with replacing XSD schemas. The model is tested by generating and visualizing GML code.

The performed implementations have confirmed that the realized GML extension is worth the effort, and adds a surplus value to GML. The new GML model is renamed GML++ here for convenience.

To the surplus value of GML++ is reckoned:

- More compact storage of geometries
- Better ensurance of constraints belonging with partitions, like no overlap of surfaces.
- GML++ links up better on topologic models of existing GIS systems (coverage in ArcInfo).
- GML 2.0 documents suffice to GML++ specification.

Concluding the development of GML++ has proven that:

- The aim of efficacy by using of chain topology is obtained
- GML++ is simple to implement and integrate in (existing) GIS software thanks to the simplicity of the available XML tools.

For the further development of GML++ the next recommendations can be reviewed:

- Realize a translation of GML++ to SVG (Scalable Vector Graphics) by means of XSLT. All SVG tools that are available then used for GML++.
- A translation for GML++ to equivalent GML 2.0 files is desired, so that the differences can be analysed thoroughly.
- Give further form to the topologic associations represented in the feature model of GML++.
- Use of the possibilities to define database types (/objects) that correspond with the GML++ types, when using the Oracle database or another relational database.
- The relation between UML and GML can be exploited deeper by research of possibilities for automatic generation of XSD”s from UML.

In the description of task (section 2.2) a few activities were determined. It is clear that there has been a slight deviation, for it is not tried to combine the package Magma/Lava with GML. Instead GML++ is integrated in the application QuickGIS.

During this graduation the development of GML has not stopped. Parallel to this project the OpenGis working parties already work on the GML 3.0 specification that amongst others will incorporate topology. This development is not taken along with in the construction of GML++. If it turns to be that GML 3.0 also chooses for the winged-edge model following of the standard is preferable of course. If not, then a redefinition of GML based on GML 3.0 is recommended.

References

[1] Tor Bernhardsen
Geographic Information Systems, An Introduction 2nd Edition
Wiley, New York, 1999

[2] G.F. Bonham-Carter
Geographic Information Systems For Geoscientist, Modeling with GIS
Pergamon, Oxford, 1994

[3] W.F. Bronsvoort, A.Noort , F.H. Post
Geometrisch modelleren; in410
TU Delft, Delft, 1997

[4] R.H.Güting
An Introduction to spatial Database Systems
Sep. 1994

[5] ISO/TC211
ISO/TC 211 Geographic information/Geomatics – Scope
Okt. 2000
http://www.statkart.no/isotc211/scope.htm

[6] ISO/TC211
ISO/TC211 Geographic information/Geomatics
2000
http://www.statkart.no/isotc211/

[7] ISO/TC211/WG2
Geographic information - Spatial Schema ISO/CD19107
Nov. 1999

[8] Martien Molenaar
An Introduction to the Theory of Spatial Object Modelling for GIS
Taylor and Francis, London, 1998

[9] Ferjan Ormeling, Menno-Jan Kraak
Kartografie, visualisatie van ruimtelijke gegevens
Delft University Press, Delft, 1999

[10] P.J.M. van Oosterom
De geo-database als spin in het web
TU Delft, Delft, 2001

[11] P.J.M. van Oosterom
Reactive Data Structures For Geographic Information Systems
Oxford University Press, Oxford, 1993

[12] Open GIS Consortium, Inc.
Geography Markup Language (GML) v1.0
Mei 2000
http://www.opengis.org/techno/specs/00-029/GML.html

[13] Open GIS Consortium, Inc.
Geography Markup Language (GML) 2.0 (OGC Document Number: 01-029)
Feb. 2001
http://www.opengis.net/gml/01-029/GML2.html

[14] Open GIS Consortium, Inc.
OpenGIS Abstract Specification, Topic 1: Feature Geometry, Version 4
Mrt. 1999

[15] Open GIS Consortium, Inc.
OpenGIS Abstract Specification, Topic 5: Features, Version 4
Mrt. 1999

[16] Open GIS Consortium, Inc.
The OpenGIS® Guide (3rd Edition) Introduction to Interoperable Geoprocessing and the OpenGIS Specification
Jun. 1998

[17] Norman Walsh
A Technical Introduction to XML
Okt. 1998
http://www.xml.com/pub/98/10/guide0.html

[18] W3C XML Working Group
Extensible Markup Language (XML) 1.0
Feb. 1998
http://www.w3.org/TR/REC-xml

[19] W3C
XML Linking Language (XLink) Version 1.0
Dec. 2000
http://www.w3.org/TR/xlink/

[20] W3C
XML Schema Part 0: Primer
Okt. 2000
http://www.w3.org/TR/xmlschema-0/

Software

[21] Altova
XML Spy
http://xmlspy.com/

[22] Apache Software Foundation
Xalan
http://xml.apache.org/xalan-j/index.html

[23] Apache Software Foundation
Xerces
http://xml.apache.org/xerces2-j/index.html

[24] bitbybit
Perspective-DB
http://vermeer.bitbybit-is.nl/NL/Products/top.html

[25] Computer Associates
CA OpenIngres
http://ca.com/products/ingres.htm

[26] ESRI
ArcInfo (v8.0.2)
http://www.esri.com/software/arcgis/arcinfo/index.html

[27] Oracle
Oracle Spatial 8i
http://otn.oracle.com/products/spatial/content.html

[28] *Perl (v 5.005_03)*
http://language.perl.com/manual/

[29] PGS
Magma/Lava
http://www.pgs.nl/

[30] SUN
Java
http://java.sun.com/

Appendix A XML specifications

In this chapter a number of XML specifications are described. The XML 1.0 specification is summarized in section A.1. Section A.2 describes the XML Schema specification. Section A.3 offers an overview of the XLink specification

A.1 XML

(*status W3C Recommendation [18]*)

A.1.1 XML terminology

A dataobject is an XML document, if it suffices to the XML syntax [17]. This is called a *well-formed* XML document. An XML document can be divided in separate dataobjects. Altogether these form the physical structure and the dataobjects are called *entities*. The logical structure is the structure of the XML document in terms of (among other) *elements*. In XML, text is structured (physical and logical) by *mark-up*. The most important forms of markup are *tags*. Tags border elements: each element starts with a *start-tag* and ends with an *end-tag*. The *content* of the element is formed by all text between the start-tag and the end-tag. Elements without content (empty-elements) can be indicated with an *empty-element-tag*. Each element has a type-name and a number of attributes. Each attribute has a name and a value. Elements are allowed to be nested (provided correctly) within an entity. A well-formed document consists of only one element: the *root-element*. An element and the elements in the content of that element are respectively called *parent* and *child(ren)* of each other.

To put restrictions to the logical structure (for a certain application), XML has *the document type declaration* mechanism at its disposal. With this the grammar can be defined for the document: the *document type definition* (DTD). It is practical to include this in an apart entity and to refer to it. When an XML document contains a document type declaration (logical) and when it suffices to the grammar the DTD imposes, it is called a *valid* XML document.

A program that is able to read XML documents and offers access to its content and structure, is called an *XML processor*. Software that make use of them, are named an *application*. XML processors have to be able to parse well-formed documents. XML processors can be divided in processors that test if XML documents suffice to the corresponding DTD (*validating processors*) and in processors that do not (*non-validating processors*)

A.1.2 XML syntax

XML has 6 different forms of markup: elements, entity references, comment, processing instructions, marked sections and document type declarations [18].

For markup the next characters are reserved:

< to indicate the begin of tags
> to indicate the end of tags
& to indicate the start of entity references
; to indicate the end of entity references

Elements

Elements start with a start-tag ***<element>***, followed by content and are closed with an end-tag ***</element>***.

Elements without any content can be indicated with an empty-element-tag ***<element/>***. Attributes of elements are included within the start-tag or empty-element-tag as a list of attributename-value pairs ***attribute="value"***, in which the values have to be between brackets.

Entity References

These serve to refer to text that is declared elsewhere (entities), that has to be inserted on the site of the reference:

&entity;

Comment

This serves to include comments in XML documents. In principle comments are neglected by XML Processors.

<!--comment-->

processing instructions

Processing instructions (PI's) are not part of XML documents, but contain information for the XML applications. PI's consist of the name of the application followed by the real data:

<?*PItarget pidata*?>

CDATA

This serves for data that is not supposed to be parsed. So all markup is neglected.

<![CDATA[*data*]]>

Document Type Declarations

These serve to specify DTDs to which XML documents need to suffice. The declaration contains a list of markup declarations or a reference to an external file with declarations.

<!DOCTYPE *root-element* [*markup-declarations*]>

<!DOCTYPE *root-element* SYSTEM "*filename*">

There are 4 types of markup declarations, namely *element type*, *attribute li*st, *entity* and *notation declarations*.

Element Type Declarations

Elements are declared with **<!ELEMENT *elementname* (*content*)>**, in which content is a list of child-elements separated by comma's or by "|". In case of commas the element has to contain all children in the given sequence and in case of "|" the element has to contain one of the given children. The children can be followed by a "**?**"or "*****" or "+", which indicates how often the child-element has to appear: at most once (**?**), at least 0 times (*****) or at least once (+).

Attribute List Declarations

These serve to declare the attributes of the element. In this per attribute the attributename, the type, the permitted value and the default value (between quotes **"** or **'**) are included.

<!ATTLIST *elementname attributes*>

Possible attributetypes are:

- **CDATA** arbitrary text
- **ID** the value must be a unique name for the identification of an element
- **IDREF** the value must be an ID value that appears in the document
- **IDREFS** the value must be a list of ID values, separated by spaces
- **ENTITY** the value must be the name of an entity (declared elsewhere)
- **ENTITIES** the value must be a list of entities, separated by spaces
- **NMTOKEN** the value must form one word (without spaces)
- **NMTOKENS** the value must be a list of nmtokens, separated by spaces
- enumerationtype is specified by a list of values that are permitted, separated by "|". The value must be in this list.

There are four possibilities for the default:

- **#REQUIRED** the attribute must always be given
- **#IMPLIED** the attribute is allowed, but does not have to be given
- **"*value*"** if the attribute is not given, it assumes this value
- **#FIXED "*value*"** if the attribute is given, the given value must be equal to this value; if it is not given, it assumes this value

Entity declarations

These serve to associate names with a text or an external file, so that Entity References can refer to it.

<!ENTITY *name* "*text*">

<!ENTITY *name* SYSTEM "*filename*">

Especially for reference to a text of the DTD there is the entity construction. References to this are only allowed within DTD according to **%*name*;**

<!ENTITY %*name* "*text*">

Notation Declarations

For the sake of completeness these declarations are necessary to specify a format type for an entity.

<!NOTATION *name* "…">

A.2 XML Schema

(*status W3C Candidate Recommendation [20]*)

Schemas are XML documents that define a vocabulary, a class of XML documents. A document that suffices to a certain schema is called an *instance* document. XML (instance) documents however can be considered apart van their schemas.

For this the XML 1.0 specification describes the DTD mechanism. Ever since, a number of alternative description techniques are developed with more powerful, more specific expressions, including *XML Schema Definition* (XSD) of the W3C.

With XSDs at least the same *vocabularies* can be described as with DTDs The advantages of XSD above DTD are:

- XML Schema uses *namespaces* to realize mixtures of different vocabularies.
- XML Schema contains a more specific and more flexible extension mechanism.
- XML Schema is able to impose more restrictions.

XSDs are XML documents and thus built from elements. In table 6 a summing up of the most important elements is shown. XML Schema makes a distinction between defining types of elements and declaring elements, with a certain element name, of a certain type.

A type can be simple or complex. A *complex type* contains sub-elements as content and/or attributes. In contrary a *simple type* contains neither sub-elements nor attributes, yet only text (CDATA). To the possibilities of simple types belong types with combinations of several types as domain (*union types*), lists of values of the same type (*list types*) and enumerations of a limited number of possible values (*enumeration types*).

The description of a complex type is called the *content model*. This can be simple (*simple content*) or complex (*complex content*), which respectively does and does not contain sub-elements.

XML Schema contains a series of predefined types (table 5).

Simple Types					
text types	binary types	numeric types	time related types	name types	XML 1.0 types
string	**base64Binary**	**integer**	**time**	**Name**	**ID**
normalizedString	**HexBinary**	**positiveInteger**	**dateTime**	**QName**	**IDREF**
token		**negativeInteger**	**duration**	**NCName**	**IDREFS**
	byte	**nonNegativeInteger**	**date**		**ENTITY**
	unsignedByte	**nonPositiveInteger**	**gMonth**	**anyURI**	**ENTITIES**
		int	**gYear**		**NOTATION**
	Boolean	**unsignedInt**	**gYearMonth**	**language**	**NMTOKEN**
		long	**gDay**		**NMTOKENS**
		unsignedLong	**gMonthDay**		
		short			
		unsignedShort			
		decimal			
		float			
		double			

table 5 XSD simple types grouped by sort

An XML Schema Definition (XSD) consists for the major part from definitions of new types (*simple* and *complex*), as sub-types of other types. These sub-types impose restrictions to permitted values (*restriction*) or extend the type with new elements or attribute (*extension*).

The complex content declares a, possibly nested, group of sub-elements (*group*) with for each element the elementname and the type. The sequence in which sub-elements are allowed to appear in an instance document can be fixed (*sequence*) or may vary (*all*). Besides, also can be indicated that only one of the sub-elements may appear in an instance document (*choice*). Further the cardinality of the sub-elements can be specified by indicating the minimum and maximum number of appearances.

The content further declares a series of attributes, by means of name and type. Each declaration can indicate whether the attribute is obliged (*required*) or can take only one value (*fixed*) or takes a predefined value (*default*) when missing. Attributes can also be grouped (*attributegroup*).

Elements of the same type, or derived from the same type, can be placed in a group to indicate that each member of the group can be used anywhere instead of the main member of the group. In an instance document, elements of a sub-type need to specify, which document it concerns. With that complex types and elements can be qualified as abstract, which means that they cannot be declared, only the derived types or synonym elements can.

XSD element	
Schema	root element
Element	declaration of an element/attribute of a certain type
Attribute	

ComplexType	definition of a new complex/simple type
SimpleType	
ComplexContent	contains the content of a new type
SimpleContent	
Group	definition of a group of elements. Defines how these elements may occur (all in the given order / all in random order / one of the listed elements)
Sequence	
All	
Choice	
Restriction	definition of the supertype of a new type. Specifies the kind of specialisation (restriction/extension)
Extention	
SubstitutionGroup	definition of a group of attributes
Include	reference to a schema defined elsewhere with the same/ a different nemespace declaration
Import	

table 6 important XSD elements

A.3 XLink

(*version 1.0, status W3C Proposal Recommendation [19]*)

The *XML Linking Language* (XLink) describes a mechanism to create and describe relations (*links*) between several addressable information-units (*resources*) in XML documents. The addressing is done with URI's.

A link consists of an enumeration of participating resources, each provided with a label, and the connections (*arcs*) between the resources that can be used. Connections consist of a starting and endpoint (*starting resource* and *ending resource*), that are identified with the *labels*. Several resources are allowed to carry the same label.

A resource is described on the spot (*local resource*) or by means of a URI a reference is made to an element elsewhere(*remote resources*).

The next three kinds of connections (arcs) are to be distinct:

- *outbound* = local starting resource and remote ending resource
- *inbound* = remote starting resource and local ending resource
- *third-party* = remote starting resource and remote ending resource

The XLink specification does not define XML elements but only a decade of (global) XML attributes that can be used in arbitrary XML documents. A link is encoded by including these attributes in different elements. The role of an element within a link is shown in the attribute type. The remaining attributes are to be subdivided into *locator attribute*, *semantic attribute*, *behaviour attributes* and *traversal attributes*.

An element with the attribute **type="extended"** contains a link. The sub-elements of that element with the attribute **type="locator"** or **type="resource"** describe the resources. In case of the type locator, it concerns a remote resource and the attribute **href** must contain a URI reference. With the attribute role another role is assigned to the resource by means of a URI. The sub-elements that contain the attribute **type="arc"** describe the connections. The starting resource is placed in the attribute **from** and the ending resource in the attribute **to**. Both contain one of the labels of the resources.

An element with the attribute **type="simple"** does not contain sub-elements with XLink attributes, but contains the attribute **href** itself, referring to a remote resource. A simple link is only able to declare an outbound link between two resources. The simple link forms a short notation for this frequently appearing link type.

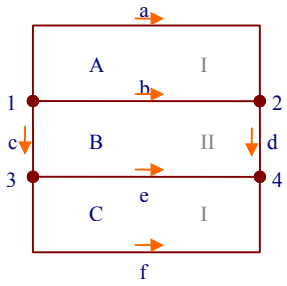
soort	Attribute	
	Type	specifies the kind of element within link
locator attribute	**Href**	refers to a resource by means of an URI
semantic attributes	**Role**	grants a role to an element by means of an URI
	Arcrole	grants a role to an arc-element by means of an URI
	Title	grants a role title to an element
behavior attributes	**Show**	describes how an application should present the ending resource when the arc is followed
	Actuate	describes when the application should follow the arc
traversal attributes	**Label**	grants a label to a resource
	From	contains the label of the starting resource(s) within a (group of) arc(s)
	To	contains the label of the ending resource(s) within a (group of) arc(s)

table 7 XLink attributes

Appendix B GML examples

B.1 example 1: GeometryComplex

This example encodes a geometric complex (without features). The encoded **GeometryComplex** consists of 4 **Point**, 6 **LineString** en 3 **Polygon** primitives.




```
<GeometryComplex gid="vb1C">
 <primitive>
  <Point id="1">
   <coord><X>30</X><Y>10</Y></coord>
  </Point>
 </primitive>
 <primitive>
  <Point id="2">
   <coord><X>40</X><Y>30</Y></coord>
  </Point>
 </primitive>
 <primitive>
  <Point id="3">
   <coord><X>20</X><Y>10</Y></coord>
  </Point>
 </primitive>
 <primitive>
  <Point id="4">
   <coord><X>40</X><Y>20</Y></coord>
  </Point>
 </primitive>
 <primitive>
  <LineString id="a" startNode="#1" endNode="#2" rightFace="#A" ncwEdge="#b" nawEdge="#c">
   <coord><X>30</X><Y>10</Y></coord><coord><X>40</X><Y>10</Y></coord>
   <coord><X>40</X><Y>40</Y></coord><coord><X>40</X><Y>30</Y></coord>
  </LineString>
 </primitive>
 <primitive>
  <LineString id="b" startNode="#1" endNode="#2" leftFace="#A" rightFace="#B" ncwEdge="#d" nawEdge="#a"/>
 </primitive>
 <primitive>
  <LineString id="c" startNode="#1" endNode="#3" leftFace="#B" ncwEdge="#f" nawEdge="#b"/>
 </primitive>
 <primitive>
  <LineString id="d" startNode="#2" endNode="#4" rightFace="#B" ncwEdge="#e" nawEdge="#a"/>
 </primitive>
 <primitive>
  <LineString id="e" startNode="#3" endNode="#4" leftFace="#B" rightFace="#C" ncwEdge="#f" nawEdge="#c"/>
 </primitive>
 <primitive>
  <LineString id="f" startNode="#3" endNode="#4" leftFace="#C" ncwEdge="#d" nawEdge="#e">
   <coord><X>20</X><Y>10</Y></coord><coord><X>10</X><Y>10</Y></coord>
   <coord><X>40</X><Y>10</Y></coord><coord><X>40</X><Y>20</Y></coord>
  </LineString>
 </primitive>
 <primitive>
  <Polygon id="A">
   <outerBoundaryIs startEdge="#a"/>
  </Polygon>
 </primitive>
 <primitive>
  <Polygon id="B">
   <outerBoundaryIs startEdge="#b"/>
  </Polygon>
 </primitive>
 <primitive>
  <Polygon id="C">
   <outerBoundaryIs startEdge="#e"/>
  </Polygon>
 </primitive>
</GeometryComplex>
```

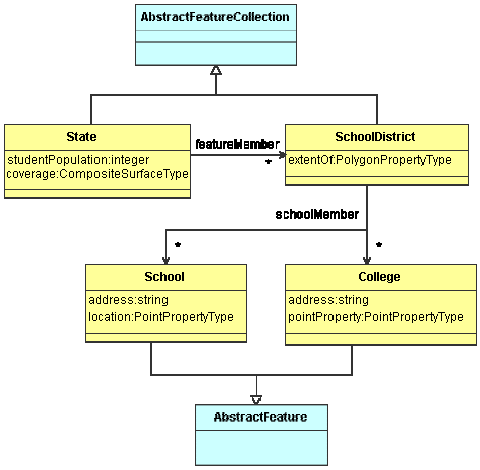
B.2 example 2: State

This example illustrates the use of *CompositeSurface* to partition the Feature **State** into **SchoolDistricts**

B.2.1 XSD

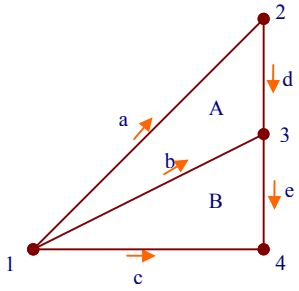
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- File: myschools.xsd  -->
<schema targetNamespace="myexamples" xmlns:ex="myexamples" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="mygml"
xmlns="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified" version="0.1">
 <annotation>
  <appinfo>myschools.xsd v2.01 2001-02</appinfo>
  <documentation xml:lang="en">
   GML schema for Schools example.
  </documentation>
 </annotation>
 <!-- import constructs from the GML Feature and Geometry schemas -->
 <import namespace="mygml" schemaLocation="myfeature.xsd"/>
 <!-- ============================================================
     global element declarations
 ============================================================== -->
 <element name="State" type="ex:StateType" substitutionGroup="gml:_FeatureCollection"/>
 <element name="SchoolDistrict" type="ex:SchoolDistrictType" substitutionGroup="gml:_FeatureCollection"/>
```

```
<element name="schoolMember" type="gml:FeatureAssociationType" substitutionGroup="gml:featureMember"/>
<element name="School" type="ex:SchoolType" substitutionGroup="gml:_Feature"/>
<element name="College" type="ex:CollegeType" substitutionGroup="gml:_Feature"/>
<element name="address" type="string"/>
<!-- ========================================================
     type definitions for state educational institutions
     ======================================================== -->
<complexType name="StateType">
 <complexContent>
  <extension base="gml:AbstractFeatureCollectionType">
   <sequence>
    <element name="studentPopulation" type="integer"/>
    <element name="coverage" type="gml:CompositeSurfaceType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
<complexType name="SchoolDistrictType">
 <complexContent>
  <extension base="gml:AbstractFeatureCollectionType">
   <sequence>
    <element ref="gml:extentOf"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
<complexType name="SchoolType">
 <complexContent>
  <extension base="gml:AbstractFeatureType">
   <sequence>
    <element ref="ex:address"/>
    <element ref="gml:location"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
<complexType name="CollegeType">
 <complexContent>
  <extension base="gml:AbstractFeatureType">
   <sequence>
    <element ref="ex:address"/>
    <element ref="gml:pointProperty"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
</schema>
```



B.2.2 XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 (http://www.xmlspy.com) -->
<!-- File: schools.xml -->
<State xmlns="myexamples" xmlns:gml="mygml" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance" xsi:schemaLocation="myexamples
myschools.xsd">
 <gml:description>
  Educational institutions with student populations exceeding 500.
 </gml:description>
 <gml:name>School districts in the North Region.</gml:name>
 <gml:boundedBy>
  <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
   <gml:coord>          <gml:X>0</gml:X><gml:Y>0</gml:Y></gml:coord>
   <gml:coord>          <gml:X>50</gml:X><gml:Y>50</gml:Y></gml:coord>
  </gml:Box>
 </gml:boundedBy>
 <gml:featureMember>
  <SchoolDistrict>
   <gml:name>District 28</gml:name>
   ...
   <schoolMember>
    ...
   </schoolMember>
   ...
   <gml:extentOf xlink:href="#B"/>
  </SchoolDistrict>
 </gml:featureMember>
 <gml:featureMember>
  <SchoolDistrict>
   <gml:name>District 32</gml:name>
   ...
   <schoolMember>
    ...
   </schoolMember>
    ...
   <gml:extentOf xlink:href="#A"/>
  </SchoolDistrict>
 </gml:featureMember>
 <studentPopulation>392620</studentPopulation>
 <coverage>
  <gml:CompositeSurface>
   <gml:GeometryComplex srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
    <gml:primitive>
     <gml:Point id="1">
      <gml:coord><gml:X>0</gml:X><gml:Y>0</gml:Y></gml:coord>
     </gml:Point>
    </gml:primitive>
    <gml:primitive>
     <gml:Point id="2">
      <gml:coord><gml:X>50</gml:X><gml:Y>50</gml:Y></gml:coord>
     </gml:Point>
    </gml:primitive>
```



```
      <gml:primitive>
       <gml:Point id="3">
         <gml:coord><gml:X>50</gml:X><gml:Y>40</gml:Y></gml:coord>
       </gml:Point>
      </gml:primitive>
      <gml:primitive>
      <gml:Point id="4">
         <gml:coord><gml:X>50</gml:X><gml:Y>0</gml:Y>              </gml:coord>
       </gml:Point>
      </gml:primitive>
      <gml:primitive>
       <gml:LineString id="a" startNode="#1" endNode="#2" rightFace="#A" ncwEdge="#d" nawEdge="#c"/>
      </gml:primitive>
      <gml:primitive>
       <gml:LineString id="b" startNode="#1" endNode="#3" leftFace="#A" rightFace="#B" ncwEdge="#e" nawEdge="#a"/>
      </gml:primitive>
      <gml:primitive>
       <gml:LineString id="c" startNode="#1" endNode="#4" leftFace="#B" ncwEdge="#e" nawEdge="#b"/>
      </gml:primitive>
      <gml:primitive>
       <gml:LineString id="d" startNode="#2" endNode="#4" rightFace="#A" ncwEdge="#b" nawEdge="#a"/>
      </gml:primitive>
      <gml:primitive>
       <gml:LineString id="e" startNode="#3" endNode="#4" rightFace="#B" ncwEdge="#c" nawEdge="#d"/>
      </gml:primitive>
      <gml:primitive/>
      <gml:primitive>
       <gml:Polygon id="A">
         <gml:outerBoundaryIs startEdge="#a"/>
       </gml:Polygon>
      </gml:primitive>
      <gml:primitive>
       <gml:Polygon id="B">
         <gml:outerBoundaryIs startEdge="#b"/>
       </gml:Polygon>
      </gml:primitive>
     </gml:GeometryComplex>
    </gml:CompositeSurface>
   </coverage>
</State>
```

B.3 example 3: facgeo

This example is the output generated by the Java application SDO2MYGML processing a fragment of the test data.

B.3.1 XSD

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- File: facgeo.xsd  -->
<schema
 targetNamespace="http://www.tdn.nl/top10test"
 xmlns="http://www.w3.org/2000/10/XMLSchema"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:gml="http://www.opengis.net/gml"
 xmlns:tdn="http://www.tdn.nl/top10test"
 elementFormDefault="qualified"
 version="0.4">

 <import namespace="http://www.opengis.net/gml" schemaLocation="feature.xsd"/>
 <element name="top10vectorobjecten"
  type="tdn:top10vectorobjectenType"
  substitutionGroup="gml:_FeatureCollection"/>

 <element name="facgeo" type="tdn:facgeo_ffeatureType" substitutionGroup="gml:_Feature"/>

 <complexType name="top10vectorobjectenType">
  <complexContent>
   <extension base="gml:AbstractFeatureCollectionType">
    <sequence>
     <element ref="gml:complexProperty" />
    </sequence>
   </extension>
  </complexContent>
 </complexType>

 <complexType name="facgeo_ffeatureType">
  <complexContent>
   <extension base="gml:AbstractFeatureType">
    <sequence>
     <element name="tdncode" type="integer" />
     <element name="area" type="integer" />
     <element name="surfaceProperty" type="surfacePropertyType" />
    </sequence>
   </extension>
  </complexContent>
 </complexType>
</schema>
```

B.3.2 XML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- File: facgeo.xml  -->
<tdn:top10vectorobjecten … >

 <gml:boundedBy>
   …
```

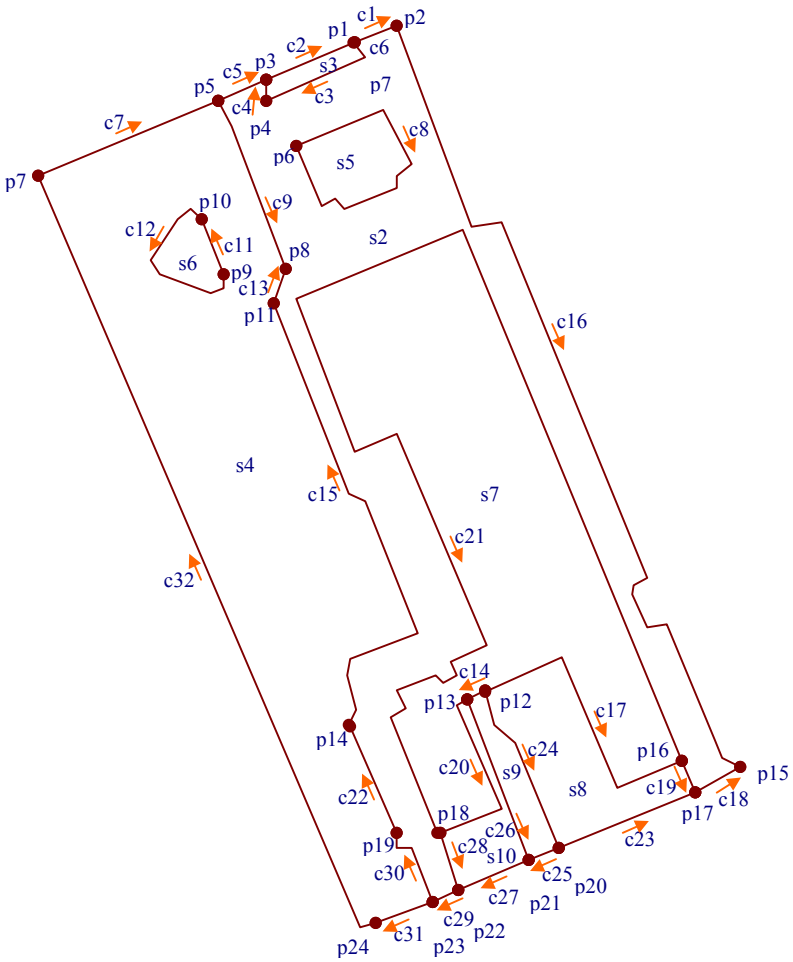
```
</gml:boundedBy>

<gml:complexProperty>
 <gml:primitive>
  <gml:Point id="p1"><gml:coord><gml:X>86324.5</gml:X><gml:Y>444745.3</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p2"><gml:coord><gml:X>86333.84</gml:X><gml:Y>444749.1</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p3"><gml:coord><gml:X>86303.12</gml:X><gml:Y>444736.3</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p4"><gml:coord><gml:X>86303.37</gml:X><gml:Y>444731.5</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p5"><gml:coord><gml:X>86291.13</gml:X><gml:Y>444731.0</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p6"><gml:coord><gml:X>86310.23</gml:X><gml:Y>444720.3</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p7"><gml:coord><gml:X>86247.48</gml:X><gml:Y>444713.0</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p8"><gml:coord><gml:X>86307.68</gml:X><gml:Y>444691.1</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p9"><gml:coord><gml:X>86292.66</gml:X><gml:Y>444688.8</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p10"><gml:coord><gml:X>86287.56</gml:X><gml:Y>444702.2</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p11"><gml:coord><gml:X>86304.37</gml:X><gml:Y>444681.9</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p12"><gml:coord><gml:X>86356.3</gml:X><gml:Y>444587.8</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p13"><gml:coord><gml:X>86352.02</gml:X><gml:Y>444585.9</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p14"><gml:coord><gml:X>86323.63</gml:X><gml:Y>444578.6</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p15"><gml:coord><gml:X>86417.91</gml:X><gml:Y>444569.3</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p16"><gml:coord><gml:X>86403.83</gml:X><gml:Y>444571.6</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p17"><gml:coord><gml:X>86406.8</gml:X><gml:Y>444563.5</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p18"><gml:coord><gml:X>86344.73</gml:X><gml:Y>444553.3</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p19"><gml:coord><gml:X>86334.24</gml:X><gml:Y>444552.6</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p20"><gml:coord><gml:X>86373.99</gml:X><gml:Y>444549.9</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p21"><gml:coord><gml:X>86366.87</gml:X><gml:Y>444547.0</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p22"><gml:coord><gml:X>86350.01</gml:X><gml:Y>444540.0</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p23"><gml:coord><gml:X>86343.45</gml:X><gml:Y>444537.2</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:Point id="p24"><gml:coord><gml:X>86326.62</gml:X><gml:Y>444530.9</gml:Y></gml:coord></gml:Point>
 </gml:primitive>
 <gml:primitive>
  <gml:LineString id="c1" startNode="#p1" endNode="#p2" rightFace="#s2" ncwEdge="#c16"/>
 </gml:primitive>
 <gml:primitive>
  <gml:LineString id="c2" startNode="#p3" endNode="#p1" rightFace="#s3" ncwEdge="#c3"/>
 </gml:primitive>
 <gml:primitive>
  <gml:LineString id="c3" startNode="#p1" endNode="#p4" leftFace="#s2" rightFace="#s3" nawEdge="#c1" ncwEdge="#c4">
   <gml:coord><gml:X>86324.5</gml:X><gml:Y>444745.3</gml:Y></gml:coord>
   <gml:coord><gml:X>86327.05</gml:X><gml:Y>444741.7</gml:Y></gml:coord>
   <gml:coord><gml:X>86303.37</gml:X><gml:Y>444731.5</gml:Y></gml:coord>
  </gml:LineString>
 </gml:primitive>
 <gml:primitive>
  <gml:LineString id="c4" startNode="#p4" endNode="#p3" leftFace="#s2" rightFace="#s3" nawEdge="#c3" ncwEdge="#c2"/>
 </gml:primitive>
 <gml:primitive>
  <gml:LineString id="c5" startNode="#p5" endNode="#p3" rightFace="#s2" ncwEdge="#c4"/>
 </gml:primitive>
 <gml:primitive>
  <gml:LineString id="c7" startNode="#p7" endNode="#p5" rightFace="#s4" ncwEdge="#c9"/>
 </gml:primitive>
 <gml:primitive>
  <gml:LineString id="c8" startNode="#p6" endNode="#p6" leftFace="#s2" rightFace="#s5" nawEdge="#c8" ncwEdge="#c8">
   <gml:coord><gml:X>86310.23</gml:X><gml:Y>444720.3</gml:Y></gml:coord>
   <gml:coord><gml:X>86331.86</gml:X><gml:Y>444729.2</gml:Y></gml:coord>
   <gml:coord><gml:X>86335.68</gml:X><gml:Y>444721.1</gml:Y></gml:coord>
   <gml:coord><gml:X>86338.23</gml:X><gml:Y>444716.3</gml:Y></gml:coord>
```

```
    <gml:coord><gml:X>86334.91</gml:X><gml:Y>444713.5</gml:Y></gml:coord>
    <gml:coord><gml:X>86335.17</gml:X><gml:Y>444709.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86321.68</gml:X><gml:Y>444704.6</gml:Y></gml:coord>
    <gml:coord><gml:X>86320.15</gml:X><gml:Y>444707.3</gml:Y></gml:coord>
    <gml:coord><gml:X>86316.34</gml:X><gml:Y>444705.8</gml:Y></gml:coord>
    <gml:coord><gml:X>86310.23</gml:X><gml:Y>444720.3</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c9" startNode="#p5" endNode="#p8" leftFace="#s2" rightFace="#s4" nawEdge="#c5" ncwEdge="#c13">
    <gml:coord><gml:X>86291.13</gml:X><gml:Y>444731.0</gml:Y></gml:coord>
    <gml:coord><gml:X>86294.44</gml:X><gml:Y>444725.2</gml:Y></gml:coord>
    <gml:coord><gml:X>86307.68</gml:X><gml:Y>444691.1</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c11" startNode="#p9" endNode="#p10" leftFace="#s6" rightFace="#s4" nawEdge="#c12" ncwEdge="#c12"/>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c12" startNode="#p10" endNode="#p9" leftFace="#s6" rightFace="#s4" nawEdge="#c11" ncwEdge="#c11">
    <gml:coord><gml:X>86287.56</gml:X><gml:Y>444702.2</gml:Y></gml:coord>
    <gml:coord><gml:X>86286.29</gml:X><gml:Y>444704.3</gml:Y></gml:coord>
    <gml:coord><gml:X>86284.0</gml:X><gml:Y>444704.8</gml:Y></gml:coord>
    <gml:coord><gml:X>86281.71</gml:X><gml:Y>444703.5</gml:Y></gml:coord>
    <gml:coord><gml:X>86277.12</gml:X><gml:Y>444695.4</gml:Y></gml:coord>
    <gml:coord><gml:X>86275.09</gml:X><gml:Y>444692.3</gml:Y></gml:coord>
    <gml:coord><gml:X>86277.12</gml:X><gml:Y>444688.5</gml:Y></gml:coord>
    <gml:coord><gml:X>86283.23</gml:X><gml:Y>444686.0</gml:Y></gml:coord>
    <gml:coord><gml:X>86289.86</gml:X><gml:Y>444684.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86292.4</gml:X><gml:Y>444685.2</gml:Y></gml:coord>
    <gml:coord><gml:X>86292.66</gml:X><gml:Y>444688.8</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c13" startNode="#p11" endNode="#p8" leftFace="#s4" rightFace="#s2" nawEdge="#c15" ncwEdge="#c9"/>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c14" startNode="#p12" endNode="#p13" leftFace="#s9" rightFace="#s7" nawEdge="#c24" ncwEdge="#c20"/>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c15" startNode="#p14" endNode="#p11" leftFace="#s4" rightFace="#s2" nawEdge="#c22" ncwEdge="#c13">
    <gml:coord><gml:X>86323.63</gml:X><gml:Y>444578.6</gml:Y></gml:coord>
    <gml:coord><gml:X>86323.08</gml:X><gml:Y>444579.9</gml:Y></gml:coord>
    <gml:coord><gml:X>86325.62</gml:X><gml:Y>444583.5</gml:Y></gml:coord>
    <gml:coord><gml:X>86322.06</gml:X><gml:Y>444591.6</gml:Y></gml:coord>
    <gml:coord><gml:X>86322.06</gml:X><gml:Y>444594.4</gml:Y></gml:coord>
    <gml:coord><gml:X>86325.37</gml:X><gml:Y>444596.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86339.62</gml:X><gml:Y>444602.3</gml:Y></gml:coord>
    <gml:coord><gml:X>86335.3</gml:X><gml:Y>444615.6</gml:Y></gml:coord>
    <gml:coord><gml:X>86327.66</gml:X><gml:Y>444633.9</gml:Y></gml:coord>
    <gml:coord><gml:X>86322.83</gml:X><gml:Y>444635.4</gml:Y></gml:coord>
    <gml:coord><gml:X>86304.37</gml:X><gml:Y>444681.9</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c16" startNode="#p2" endNode="#p15" rightFace="#s2" ncwEdge="#c18">
    <gml:coord><gml:X>86333.84</gml:X><gml:Y>444749.1</gml:Y></gml:coord>
    <gml:coord><gml:X>86338.48</gml:X><gml:Y>444737.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86352.87</gml:X><gml:Y>444700.3</gml:Y></gml:coord>
    <gml:coord><gml:X>86360.25</gml:X><gml:Y>444701.4</gml:Y></gml:coord>
    <gml:coord><gml:X>86382.14</gml:X><gml:Y>444648.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86395.26</gml:X><gml:Y>444615.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86392.2</gml:X><gml:Y>444612.9</gml:Y></gml:coord>
    <gml:coord><gml:X>86392.2</gml:X><gml:Y>444610.3</gml:Y></gml:coord>
    <gml:coord><gml:X>86395.51</gml:X><gml:Y>444603.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86400.09</gml:X><gml:Y>444604.2</gml:Y></gml:coord>
    <gml:coord><gml:X>86411.04</gml:X><gml:Y>444577.5</gml:Y></gml:coord>
    <gml:coord><gml:X>86414.35</gml:X><gml:Y>444571.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86417.91</gml:X><gml:Y>444569.3</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c17" startNode="#p16" endNode="#p12" leftFace="#s8" rightFace="#s7" nawEdge="#c19" ncwEdge="#c14">
    <gml:coord><gml:X>86403.83</gml:X><gml:Y>444571.6</gml:Y></gml:coord>
    <gml:coord><gml:X>86387.98</gml:X><gml:Y>444565.1</gml:Y></gml:coord>
    <gml:coord><gml:X>86374.99</gml:X><gml:Y>444596.3</gml:Y></gml:coord>
    <gml:coord><gml:X>86356.3</gml:X><gml:Y>444587.8</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c18" startNode="#p17" endNode="#p15" leftFace="#s2" nawEdge="#c19"/>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c19" startNode="#p16" endNode="#p17" leftFace="#s2" rightFace="#s8" nawEdge="#c21" ncwEdge="#c23"/>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c20" startNode="#p13" endNode="#p18" leftFace="#s10" rightFace="#s7" nawEdge="#c26" ncwEdge="#c21">
    <gml:coord><gml:X>86352.02</gml:X><gml:Y>444585.9</gml:Y></gml:coord>
    <gml:coord><gml:X>86349.16</gml:X><gml:Y>444584.6</gml:Y></gml:coord>
    <gml:coord><gml:X>86359.88</gml:X><gml:Y>444559.6</gml:Y></gml:coord>
    <gml:coord><gml:X>86344.73</gml:X><gml:Y>444553.3</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c21" startNode="#p16" endNode="#p18" leftFace="#s7" rightFace="#s2" nawEdge="#c17" ncwEdge="#c28">
    <gml:coord><gml:X>86403.83</gml:X><gml:Y>444571.6</gml:Y></gml:coord>
    <gml:coord><gml:X>86350.53</gml:X><gml:Y>444699.8</gml:Y></gml:coord>
    <gml:coord><gml:X>86309.47</gml:X><gml:Y>444682.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86324.77</gml:X><gml:Y>444645.9</gml:Y></gml:coord>
    <gml:coord><gml:X>86335.1</gml:X><gml:Y>444650.2</gml:Y></gml:coord>
    <gml:coord><gml:X>86356.52</gml:X><gml:Y>444598.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86347.86</gml:X><gml:Y>444595.1</gml:Y></gml:coord>
    <gml:coord><gml:X>86349.41</gml:X><gml:Y>444591.3</gml:Y></gml:coord>
```

```
    <gml:coord><gml:X>86345.58</gml:X><gml:Y>444589.8</gml:Y></gml:coord>
    <gml:coord><gml:X>86344.5</gml:X><gml:Y>444592.3</gml:Y></gml:coord>
    <gml:coord><gml:X>86334.8</gml:X><gml:Y>444588.3</gml:Y></gml:coord>
    <gml:coord><gml:X>86336.7</gml:X><gml:Y>444583.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86332.76</gml:X><gml:Y>444582.1</gml:Y></gml:coord>
    <gml:coord><gml:X>86344.73</gml:X><gml:Y>444553.3</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c22" startNode="#p19" endNode="#p14" leftFace="#s4" rightFace="#s2" nawEdge="#c30" ncwEdge="#c15"/>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c23" startNode="#p17" endNode="#p20" rightFace="#s8" ncwEdge="#c24">
    <gml:coord><gml:X>86406.8</gml:X><gml:Y>444563.5</gml:Y></gml:coord>
    <gml:coord><gml:X>86397.39</gml:X><gml:Y>444559.4</gml:Y></gml:coord>
    <gml:coord><gml:X>86373.99</gml:X><gml:Y>444549.9</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c24" startNode="#p12" endNode="#p20" leftFace="#s8" rightFace="#s9" nawEdge="#c17" ncwEdge="#c25">
    <gml:coord><gml:X>86356.3</gml:X><gml:Y>444587.8</gml:Y></gml:coord>
    <gml:coord><gml:X>86358.72</gml:X><gml:Y>444579.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86363.81</gml:X><gml:Y>444574.8</gml:Y></gml:coord>
    <gml:coord><gml:X>86373.99</gml:X><gml:Y>444549.9</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c25" startNode="#p20" endNode="#p21" rightFace="#s9" ncwEdge="#c26"/>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c26" startNode="#p13" endNode="#p21" leftFace="#s9" rightFace="#s10" nawEdge="#c14" ncwEdge="#c27"/>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c27" startNode="#p21" endNode="#p22" rightFace="#s10" ncwEdge="#c28"/>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c28" startNode="#p18" endNode="#p22" leftFace="#s10" rightFace="#s2" nawEdge="#c20" ncwEdge="#c29">
    <gml:coord><gml:X>86344.73</gml:X><gml:Y>444553.3</gml:Y></gml:coord>
    <gml:coord><gml:X>86347.01</gml:X><gml:Y>444547.1</gml:Y></gml:coord>
    <gml:coord><gml:X>86350.01</gml:X><gml:Y>444540.0</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c29" startNode="#p22" endNode="#p23" rightFace="#s2" ncwEdge="#c30"/>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c30" startNode="#p23" endNode="#p19" leftFace="#s4" rightFace="#s2" nawEdge="#c31" ncwEdge="#c22">
    <gml:coord><gml:X>86343.45</gml:X><gml:Y>444537.2</gml:Y></gml:coord>
    <gml:coord><gml:X>86337.84</gml:X><gml:Y>444550.6</gml:Y></gml:coord>
    <gml:coord><gml:X>86335.05</gml:X><gml:Y>444550.6</gml:Y></gml:coord>
    <gml:coord><gml:X>86334.24</gml:X><gml:Y>444552.6</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c31" startNode="#p23" endNode="#p24" rightFace="#s4" ncwEdge="#c32">
    <gml:coord><gml:X>86343.45</gml:X><gml:Y>444537.2</gml:Y></gml:coord>
    <gml:coord><gml:X>86335.27</gml:X><gml:Y>444533.7</gml:Y></gml:coord>
    <gml:coord><gml:X>86326.62</gml:X><gml:Y>444530.9</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:LineString id="c32" startNode="#p24" endNode="#p7" rightFace="#s4" ncwEdge="#c7">
    <gml:coord><gml:X>86326.62</gml:X><gml:Y>444530.9</gml:Y></gml:coord>
    <gml:coord><gml:X>86325.09</gml:X><gml:Y>444531.4</gml:Y></gml:coord>
    <gml:coord><gml:X>86316.18</gml:X><gml:Y>444552.0</gml:Y></gml:coord>
    <gml:coord><gml:X>86293.65</gml:X><gml:Y>444605.9</gml:Y></gml:coord>
    <gml:coord><gml:X>86279.64</gml:X><gml:Y>444637.4</gml:Y></gml:coord>
    <gml:coord><gml:X>86247.48</gml:X><gml:Y>444713.0</gml:Y></gml:coord>
   </gml:LineString>
  </gml:primitive>
  <gml:primitive>
   <gml:Polygon id="s2">
    <outerBoundaryIs startEdge="#c4"/>
    <innerBoundaryIs startEdge="#c8"/>
   </gml:Polygon>
  </gml:primitive>
  <gml:primitive>
   <gml:Polygon id="s3">
    <outerBoundaryIs startEdge="#c2"/>
   </gml:Polygon>
  </gml:primitive>
  <gml:primitive>
   <gml:Polygon id="s4">
    <outerBoundaryIs startEdge="#c9"/>
    <innerBoundaryIs startEdge="#c11"/>
   </gml:Polygon>
  </gml:primitive>
  <gml:primitive>
   <gml:Polygon id="s5">
    <outerBoundaryIs startEdge="#c8"/>
   </gml:Polygon>
  </gml:primitive>
  <gml:primitive>
   <gml:Polygon id="s6">
    <outerBoundaryIs startEdge="#c11"/>
   </gml:Polygon>
  </gml:primitive>
  <gml:primitive>
   <gml:Polygon id="s7">
    <outerBoundaryIs startEdge="#c14"/>
   </gml:Polygon>
  </gml:primitive>
  <gml:primitive>
```

```
    <gml:Polygon id="s8">
      <outerBoundaryIs startEdge="#c17"/>
    </gml:Polygon>
   </gml:primitive>
   <gml:primitive>
    <gml:Polygon id="s9">
      <outerBoundaryIs startEdge="#c14"/>
    </gml:Polygon>
   </gml:primitive>
   <gml:primitive>
    <gml:Polygon id="s10">
      <outerBoundaryIs startEdge="#c20"/>
    </gml:Polygon>
   </gml:primitive>
  </gml:complexProperty>
  <tdn:featureMember>
   <tdn:facgeo_ffeature fid="f2">
    <tdn:tdncode>5263</tdn:tdncode>
    <tdn:area>4903</tdn:area>
    <gml:surfaceProperty href="#s2"/>
   </tdn:facgeo_ffeature>
  </tdn:featureMember>
  <tdn:featureMember>
   <tdn:facgeo_ffeature fid="f3">
    <tdn:tdncode>5263</tdn:tdncode>
    <tdn:area>108</tdn:area>
    <gml:surfaceProperty href="#s3"/>
   </tdn:facgeo_ffeature>
  </tdn:featureMember>
  <tdn:featureMember>
   <tdn:facgeo_ffeature fid="f4">
    <tdn:tdncode>5213</tdn:tdncode>
    <tdn:area>6798</tdn:area>
    <gml:surfaceProperty href="#s4"/>
   </tdn:facgeo_ffeature>
  </tdn:featureMember>
  <tdn:featureMember>
   <tdn:facgeo_ffeature fid="f5">
    <tdn:tdncode>5263</tdn:tdncode>
    <tdn:area>424</tdn:area>
    <gml:surfaceProperty href="#s5"/>
   </tdn:facgeo_ffeature>
  </tdn:featureMember>
  <tdn:featureMember>
   <tdn:facgeo_ffeature fid="f6">
    <tdn:tdncode>6113</tdn:tdncode>
    <tdn:area>223</tdn:area>
    <gml:surfaceProperty href="#s6"/>
   </tdn:facgeo_ffeature>
  </tdn:featureMember>
  <tdn:featureMember>
   <tdn:facgeo_ffeature fid="f7">
    <tdn:tdncode>1013</tdn:tdncode>
    <tdn:area>5196</tdn:area>
    <gml:surfaceProperty href="#s7"/>
   </tdn:facgeo_ffeature>
  </tdn:featureMember>
  <tdn:featureMember>
   <tdn:facgeo_ffeature fid="f8">
    <tdn:tdncode>5263</tdn:tdncode>
    <tdn:area>965</tdn:area>
    <gml:surfaceProperty href="#s8"/>
   </tdn:facgeo_ffeature>
  </tdn:featureMember>
  <tdn:featureMember>
   <tdn:facgeo_ffeature fid="f9">
    <tdn:tdncode>3433</tdn:tdncode>
    <tdn:area>270</tdn:area>
    <gml:surfaceProperty href="#s9"/>
   </tdn:facgeo_ffeature>
  </tdn:featureMember>
  <tdn:featureMember>
   <tdn:facgeo_ffeature fid="f10">
    <tdn:tdncode>5213</tdn:tdncode>
    <tdn:area>335</tdn:area>
    <gml:surfaceProperty href="#s10"/>
   </tdn:facgeo_ffeature>
  </tdn:featureMember>
</tdn:top10vectorobjecten>
```



Appendix C Schemas

De XSDs presented in this chapter are meant to replace the XSD;s of GML 2.0 XSDs. These schemas are established by adapting the original GML 2.0 schemas.

C.1 Encoding GML++

Geometry.xsd is revisited to make the encoding of complex geometry possible.

GML++ provides support for the encoding of complex geometries. The topology model winged-edge topology is incorporated in GML++ by adding topology attributes to the geometry types. These attributes contain URI references, which should point to other geometry elements.

C.1.1 Encoding complex geometry

A complex geometry can be viewed as a special case of a heterogeneous geometry collection. Instead of being consisted of **geometryMember** elements, the **ComplexGeometry** element contains primitive elements. Each primitive element contains a geometry element. These geometry elements however must contain values for their topologic attributes. These values are URI references to their adjacent primitives, according to the winged-edge principle. Examples are presented in Appendix B.

Complex geometry elements can be used anywhere where a geometry is expected. Furthermore curves and surfaces can be encoded as **Composite_Curve** and **Composite_Surface**, which contain a **GeometryComplex**.

C.1.2 Encoding features with complex geometry

GML++ also provides complex related geometry property elements (in features.xsd). The element **complexGeometryProperty** is intended to be used within feature collections (outside its members). Each feature member should contain a geometry property with a URI reference (**href**) which points to one of the primitives of the complex geometry. GML++ however doesn't (can't) restrict references to these primitives. (If this is preferred, the use of Xpath might realize this in future GML updates.)

C.2 geometry.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 (http://www.xmlspy.com) by cc -->
<!--  File: geometry.xsd  -->
<schema targetNamespace="mygml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="mygml"
xmlns="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified" version="0.1">
 <annotation>
  <appinfo>geometry.xsd v2.06 2001-02</appinfo>
  <documentation xml:lang="en">
   GML Geometry schema. Copyright (c) 2001 OGC, All Rights Reserved.
  </documentation>
 </annotation>
 <!-- bring in the XLink attributes -->
 <import namespace="http://www.w3.org/1999/xlink"
schemaLocation="xlinks.xsd"/>
 <!--
================================================================
====
     global declarations

================================================================
========= -->
 <element name="_Geometry" type="gml:AbstractGeometryType"
abstract="true"/>
 <element name="_GeometryCollection"
type="gml:GeometryCollectionType" abstract="true"/>
 <element name="geometryMember"
type="gml:GeometryAssociationType"/>
 <element name="primitive" type="gml:GeometryAssociationType"/>
 <element name="GeometryComplex" type="gml:GeometryComplexType"/>
 <!-- primitive geometry elements -->
 <element name="Point" type="gml:PointType"
substitutionGroup="gml:_Geometry"/>
 <element name="_Curve" type="gml:AbstractCurveType"
substitutionGroup="gml:_Geometry"/>
 <element name="_Surface" type="gml:AbstractSurfaceType"
substitutionGroup="gml:_Geometry"/>
 <element name="Box" type="gml:BoxType"/>
 <!-- derived geometry elements -->
 <element name="CompositeCurve" type="gml:CompositeCurveType"
substitutionGroup="gml:_Curve"/>
 <element name="CompositeSurface" type="gml:CompositeSurfaceType"
substitutionGroup="gml:_Surface"/>
 <element name="LineString" type="gml:LineStringType"
substitutionGroup="gml:_Geometry"/>
 <element name="LinearRing" type="gml:LinearRingType"
substitutionGroup="gml:_Geometry"/>
 <element name="Polygon" type="gml:PolygonType"
substitutionGroup="gml:_Geometry"/>
 <!-- aggregate geometry elements -->
 <element name="MultiGeometry" type="gml:GeometryCollectionType"/>
 <element name="MultiPoint" type="gml:MultiPointType"
substitutionGroup="gml:_Geometry"/>
 <element name="MultiCurve" type="gml:MultiCurveType"
substitutionGroup="gml:_Geometry"/>
 <element name="MultiSurface" type="gml:MultiSurfaceType"
substitutionGroup="gml:_Geometry"/>
 <element name="MultiLineString" type="gml:MultiLineStringType"
substitutionGroup="gml:_Geometry"/>
 <element name="MultiPolygon" type="gml:MultiPolygonType"
substitutionGroup="gml:_Geometry"/>
 <!-- coordinate elements -->
 <element name="coord" type="gml:CoordType"/>
 <element name="coordinates" type="gml:CoordinatesType"/>
 <!-- this attribute gives the location where an element is defined -->
 <attribute name="remoteSchema" type="uriReference"/>
 <!--
================================================================
====
     abstract supertypes

================================================================
========= -->
 <complexType name="AbstractGeometryType" abstract="true">
  <annotation>
   <documentation>
    All geometry elements are derived from this abstract supertype;
    a geometry element may have an identifying attribute ('gid').
    It may be associated with a spatial reference system.
   </documentation>
  </annotation>
  <attribute name="gid" type="ID" use="optional"/>
  <attribute name="srsName" type="uriReference" use="optional"/>
 </complexType>
 <complexType name="AbstractGeometryCollectionBaseType"
abstract="true">
  <annotation>
   <documentation>
    This abstract base type for geometry collections just makes the
    srsName attribute mandatory.
   </documentation>
```

```
  </annotation>
  <complexContent>
   <restriction base="gml:AbstractGeometryType">
    <attribute name="gid" type="ID" use="optional"/>
    <attribute name="srsName" type="uriReference" use="required"/>
   </restriction>
  </complexContent>
 </complexType>
 <attributeGroup name="AssociationAttributeGroup">
  <annotation>
   <documentation>
    These attributes can be attached to any element, thus allowing it
    to act as a pointer. The 'remoteSchema' attribute allows an element
    that carries link attributes to indicate that the element is declared
    in a remote schema rather than by the schema that constrains the
    current document instance.
   </documentation>
  </annotation>
  <attributeGroup ref="xlink:simpleLink"/>
  <attribute ref="gml:remoteSchema" use="optional"/>
 </attributeGroup>
 <complexType name="GeometryAssociationType">
  <annotation>
   <documentation>
    An instance of this type (e.g. a geometryMember) can either
    enclose or point to a primitive geometry element. When serving
    as a simple link that references a remote geometry instance,
    the value of the gml:remoteSchema attribute can be used to
    locate a schema fragment that constrains the target instance.
   </documentation>
  </annotation>
  <sequence>
   <element ref="gml:_Geometry" minOccurs="0"/>
  </sequence>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
 </complexType>
 <complexType name="AbstractPointType">
  <complexContent>
   <extension base="gml:AbstractGeometryType"/>
  </complexContent>
 </complexType>
 <complexType name="AbstractCurveType" abstract="true">
  <complexContent>
   <extension base="gml:AbstractGeometryType">
    <attributeGroup ref="gml:EdgeAttributeGroup"/>
   </extension>
  </complexContent>
 </complexType>
 <complexType name="AbstractSurfaceType" abstract="true">
  <complexContent>
   <extension base="gml:AbstractGeometryType">
    <sequence>
     <element name="outerBoundaryIs" type="gml:AbstractRingType"/>
     <element name="innerBoundaryIs" type="gml:AbstractRingType"
minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
 <complexType name="AbstractRingType">
  <complexContent>
   <extension base="gml:AbstractGeometryType">
    <attributeGroup ref="gml:RingAttributeGroup"/>
   </extension>
  </complexContent>
 </complexType>
 <!--
=============================================================
====
     primitive geometry types

=============================================================
========= -->
 <complexType name="PointType">
  <annotation>
   <documentation>
    A Point is defined by a single coordinate tuple.
   </documentation>
  </annotation>
  <complexContent>
   <extension base="gml:AbstractPointType">
    <sequence>
     <choice>
      <element ref="gml:coord"/>
      <element ref="gml:coordinates"/>
     </choice>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
 <complexType name="LineStringType">
  <annotation>
   <documentation>
    A LineString is defined by two or more coordinate tuples, with
    linear interpolation between them.
   </documentation>
  </annotation>
  <complexContent>
   <extension base="gml:AbstractCurveType">
    <sequence>
     <choice>
      <element ref="gml:coord" minOccurs="2" maxOccurs="unbounded"/>
      <element ref="gml:coordinates"/>
     </choice>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
 <complexType name="LinearRingType">
  <annotation>
   <documentation>
    A LinearRing is defined by four or more coordinate tuples, with
    linear interpolation between them; the first and last coordinates
    must be coincident.
   </documentation>
  </annotation>
  <complexContent>
   <extension base="gml:AbstractRingType"/>
  </complexContent>
 </complexType>
 <complexType name="BoxType">
  <annotation>
   <documentation>
    The Box structure defines an extent using a pair of coordinate tuples.
   </documentation>
  </annotation>
  <complexContent>
   <extension base="gml:AbstractGeometryType">
    <sequence>
     <choice>
      <element ref="gml:coord" minOccurs="2" maxOccurs="2"/>
      <element ref="gml:coordinates"/>
     </choice>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
 <complexType name="PolygonType">
  <annotation>
   <documentation>
    A Polygon is defined by an outer boundary and zero or more inner
    boundaries which are in turn defined by LinearRings.
   </documentation>
  </annotation>
  <complexContent>
   <extension base="gml:AbstractSurfaceType"/>
  </complexContent>
 </complexType>
 <!--
=============================================================
====
     aggregate geometry types

=============================================================
========= -->
 <complexType name="GeometryCollectionType">
  <annotation>
   <documentation>
    A geometry collection must include one or more geometries, referenced
    through geometryMember elements. User-defined geometry collections
    that accept GML geometry classes as members must instantiate--or
    derive from--this type.
   </documentation>
  </annotation>
  <complexContent>
   <extension base="gml:AbstractGeometryCollectionBaseType">
    <sequence>
     <element ref="gml:geometryMember" maxOccurs="unbounded"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
 <complexType name="MultiPointType">
  <annotation>
   <documentation>
    A MultiPoint is defined by one or more Points, referenced through
    pointMember elements.
   </documentation>
  </annotation>
  <complexContent>
   <restriction base="gml:GeometryCollectionType">
    <sequence>
     <element name="pointMember" maxOccurs="unbounded">
      <complexType>
       <sequence>
        <element ref="gml:Point"/>
       </sequence>
      </complexType>
     </element>
    </sequence>
   </restriction>
  </complexContent>
 </complexType>
 <complexType name="MultiCurveType">
  <complexContent>
   <restriction base="gml:GeometryCollectionType">
    <sequence>
     <element name="curveMember" maxOccurs="unbounded">
      <complexType>
       <sequence>
        <element ref="gml:_Curve"/>
       </sequence>
      </complexType>
     </element>
    </sequence>
   </restriction>
```

```
  </complexContent>
 </complexType>
 <complexType name="MultiSurfaceType">
  <complexContent>
   <restriction base="gml:GeometryCollectionType">
    <sequence>
     <element name="surfaceMember" maxOccurs="unbounded">
      <complexType>
       <sequence>
        <element ref="gml:_Surface"/>
       </sequence>
      </complexType>
     </element>
    </sequence>
   </restriction>
  </complexContent>
 </complexType>
 <complexType name="MultiLineStringType">
  <annotation>
   <documentation>
    A MultiLineString is defined by one or more LineStrings, referenced
    through lineStringMember elements.
   </documentation>
  </annotation>
  <complexContent>
   <restriction base="gml:MultiCurveType">
    <sequence>
     <element name="lineStringMember" maxOccurs="unbounded">
      <complexType>
       <sequence>
        <element ref="gml:LineString"/>
       </sequence>
      </complexType>
     </element>
    </sequence>
   </restriction>
  </complexContent>
 </complexType>
 <complexType name="MultiPolygonType">
  <annotation>
   <documentation>
    A MultiPolygon is defined by one or more Polygons, referenced through
    polygonMember elements.
   </documentation>
  </annotation>
  <complexContent>
   <restriction base="gml:MultiLineStringType">
    <sequence>
     <element name="polygonMember" maxOccurs="unbounded">
      <complexType>
       <sequence>
        <element ref="gml:Polygon"/>
       </sequence>
      </complexType>
     </element>
    </sequence>
   </restriction>
  </complexContent>
 </complexType>
 <complexType name="GeometryComplexType">
  <complexContent>
   <extension base="gml:AbstractGeometryCollectionBaseType">
    <sequence>
     <element ref="gml:primitive" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
 <complexType name="CompositeCurveType">
  <complexContent>
   <extension base="gml:AbstractCurveType">
    <sequence>
     <element ref="gml:GeometryComplex"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
 <complexType name="CompositeSurfaceType">
  <complexContent>
   <extension base="gml:AbstractSurfaceType">
    <sequence>
     <element ref="gml:GeometryComplex"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
 <!--
================================================================
====
     There are two ways to represent coordinates: (1) as a sequence
     of <coord> elements that encapsulate tuples, or (2) using a
     single <coordinates> string.

================================================================
========= -->
 <complexType name="CoordType">
  <annotation>
   <documentation>
    Represents a coordinate tuple in one, two, or three dimensions.
   </documentation>
  </annotation>
  <sequence>
   <element name="X" type="decimal"/>
   <element name="Y" type="decimal" minOccurs="0"/>
   <element name="Z" type="decimal" minOccurs="0"/>
  </sequence>
 </complexType>
 <complexType name="CoordinatesType">
  <annotation>
   <documentation>
    Coordinates can be included in a single string, but there is no
    facility for validating string content. The value of the 'cs' attribute
    is the separator for coordinate values, and the value of the 'ts'
    attribute gives the tuple separator (a single space by default); the
    default values may be changed to reflect local usage.
   </documentation>
  </annotation>
  <simpleContent>
   <extension base="string">
    <attribute name="decimal" type="string" use="default" value="."/>
    <attribute name="cs" type="string" use="default" value=","/>
    <attribute name="ts" type="string" use="default" value="&#x20;"/>
   </extension>
  </simpleContent>
 </complexType>
 <attributeGroup name="NodeAttributeGroup">
  <attribute name="containingFace" type="uriReference" use="optional"/>
 </attributeGroup>
 <attributeGroup name="EdgeAttributeGroup">
  <attribute name="startNode" type="uriReference" use="optional"/>
  <attribute name="endNode" type="uriReference" use="optional"/>
  <attribute name="leftFace" type="uriReference" use="optional"/>
  <attribute name="rightFace" type="uriReference" use="optional"/>
  <attribute name="ncwEdge" type="uriReference" use="optional"/>
  <attribute name="nawEdge" type="uriReference" use="optional"/>
 </attributeGroup>
 <attributeGroup name="RingAttributeGroup">
  <attribute name="startEdge" type="uriReference" use="optional"/>
 </attributeGroup>
</schema>
```

C.3 feature.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 (http://www.xmlspy.com) -->
<!-- File: feature.xsd  -->
<schema targetNamespace="mygml" xmlns:gml="mygml"
xmlns="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified" version="0.1">
 <annotation>
  <appinfo>feature.xsd v2.06 2001-02</appinfo>
  <documentation xml:lang="en">
   GML Feature schema. Copyright (c) 2001 OGC, All Rights Reserved.
  </documentation>
 </annotation>
 <!-- include constructs from the GML Geometry schema -->
 <include schemaLocation="mygeometry.xsd"/>
 <!--
================================================================
====
     global declarations

================================================================
========= -->
 <element name="_Feature" type="gml:AbstractFeatureType"
abstract="true"/>
 <element name="_FeatureCollection"
type="gml:AbstractFeatureCollectionType" abstract="true"
substitutionGroup="gml:_Feature"/>
 <element name="featureMember" type="gml:FeatureAssociationType"/>
 <!-- some basic geometric properties of features -->
 <element name="_geometryProperty" type="gml:GeometryPropertyType"
abstract="true"/>
 <element name="geometryProperty" type="gml:GeometryPropertyType"/>
 <element name="boundedBy" type="gml:BoundingShapeType"/>
 <element name="pointProperty" type="gml:PointPropertyType"
substitutionGroup="gml:_geometryProperty"/>
 <element name="curveProperty" type="gml:CurvePropertyType"
substitutionGroup="gml:_geometryProperty"/>
 <element name="surfaceProperty" type="gml:SurfacePropertyType"
substitutionGroup="gml:_geometryProperty"/>
 <element name="lineStringProperty" type="gml:LineStringPropertyType"
substitutionGroup="gml:curveProperty"/>
 <element name="polygonProperty" type="gml:PolygonPropertyType"
substitutionGroup="gml:surfaceProperty"/>
 <element name="multiPointProperty" type="gml:MultiPointPropertyType"
substitutionGroup="gml:_geometryProperty"/>
 <element name="multiCurveProperty" type="gml:MultiCurvePropertyType"
substitutionGroup="gml:_geometryProperty"/>
 <element name="multiSurfaceProperty"
type="gml:MultiSurfacePropertyType"
substitutionGroup="gml:_geometryProperty"/>
 <element name="multiLineStringProperty"
type="gml:MultiLineStringPropertyType"
substitutionGroup="gml:multiCurveProperty"/>
 <element name="multiPolygonProperty"
type="gml:MultiPolygonPropertyType"
substitutionGroup="gml:multiSurfaceProperty"/>
```

```
  <element name="multiGeometryProperty"
type="gml:MultiGeometryPropertyType"
substitutionGroup="gml:_geometryProperty"/>
  <element name="complexProperty" type="gml:ComplexPropertyType"
substitutionGroup="gml:_geometryProperty"/>
  <!-- common aliases for geometry properties -->
  <element name="location" type="gml:PointPropertyType"
substitutionGroup="gml:pointProperty"/>
  <element name="centerOf" type="gml:PointPropertyType"
substitutionGroup="gml:pointProperty"/>
  <element name="position" type="gml:PointPropertyType"
substitutionGroup="gml:pointProperty"/>
  <element name="extentOf" type="gml:PolygonPropertyType"
substitutionGroup="gml:polygonProperty"/>
  <element name="coverage" type="gml:PolygonPropertyType"
substitutionGroup="gml:polygonProperty"/>
  <element name="edgeOf" type="gml:LineStringPropertyType"
substitutionGroup="gml:lineStringProperty"/>
  <element name="centerLineOf" type="gml:LineStringPropertyType"
substitutionGroup="gml:lineStringProperty"/>
  <element name="multiLocation" type="gml:MultiPointPropertyType"
substitutionGroup="gml:multiPointProperty"/>
  <element name="multiCenterOf" type="gml:MultiPointPropertyType"
substitutionGroup="gml:multiPointProperty"/>
  <element name="multiPosition" type="gml:MultiPointPropertyType"
substitutionGroup="gml:multiPointProperty"/>
  <element name="multiCenterLineOf"
type="gml:MultiLineStringPropertyType"
substitutionGroup="gml:multiLineStringProperty"/>
  <element name="multiEdgeOf" type="gml:MultiLineStringPropertyType"
substitutionGroup="gml:multiLineStringProperty"/>
  <element name="multiCoverage" type="gml:MultiPolygonPropertyType"
substitutionGroup="gml:multiPolygonProperty"/>
  <element name="multiExtentOf" type="gml:MultiPolygonPropertyType"
substitutionGroup="gml:multiPolygonProperty"/>
  <!-- common feature descriptors -->
  <element name="description" type="string"/>
  <element name="name" type="string"/>
  <!--
================================================================
====
      abstract supertypes

================================================================
========= -->
  <complexType name="AbstractFeatureType" abstract="true">
    <annotation>
      <documentation>
        An abstract feature provides a set of common properties. A concrete
        feature type must derive from this type and specify additional
        properties in an application schema. A feature may optionally
        possess an identifying attribute ('fid').
      </documentation>
    </annotation>
    <sequence>
      <element ref="gml:description" minOccurs="0"/>
      <element ref="gml:name" minOccurs="0"/>
      <element ref="gml:boundedBy" minOccurs="0"/>
      <!-- additional properties must be specified in an application schema -->
    </sequence>
    <attribute name="fid" type="ID" use="optional"/>
  </complexType>
  <complexType name="AbstractFeatureCollectionBaseType"
abstract="true">
    <annotation>
      <documentation>
        This abstract base type just makes the boundedBy element mandatory
        for a feature collection.
      </documentation>
    </annotation>
    <complexContent>
      <restriction base="gml:AbstractFeatureType">
        <sequence>
          <element ref="gml:description" minOccurs="0"/>
          <element ref="gml:name" minOccurs="0"/>
          <element ref="gml:boundedBy"/>
        </sequence>
        <attribute name="fid" type="ID" use="optional"/>
      </restriction>
    </complexContent>
  </complexType>
  <complexType name="AbstractFeatureCollectionType" abstract="true">
    <annotation>
      <documentation>
        A feature collection contains zero or more featureMember elements.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="gml:AbstractFeatureCollectionBaseType">
        <sequence>
          <element ref="gml:featureMember" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GeometryPropertyType">
    <annotation>
      <documentation>
        A simple geometry property encapsulates a geometry element.
        Alternatively, it can function as a pointer (simple-type link)
        that refers to a remote geometry element.
      </documentation>
    </annotation>
    <sequence minOccurs="0">
      <element ref="gml:_Geometry"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
  <complexType name="FeatureAssociationType">
    <annotation>
      <documentation>
        An instance of this type (e.g. a featureMember) can either
        enclose or point to a feature (or feature collection); this
        type can be restricted in an application schema to allow only
        specified features as valid participants in the association.
        When serving as a simple link that references a remote feature
        instance, the value of the gml:remoteSchema attribute can be
        used to locate a schema fragment that constrains the target
        instance.
      </documentation>
    </annotation>
    <sequence minOccurs="0">
      <element ref="gml:_Feature"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
  <complexType name="BoundingShapeType">
    <annotation>
      <documentation>
        Bounding shapes--a Box or a null element are currently allowed.
      </documentation>
    </annotation>
    <sequence>
      <choice>
        <element ref="gml:Box"/>
        <element name="null" type="gml:NullType"/>
      </choice>
    </sequence>
  </complexType>
  <!--
================================================================
====
      geometry properties

================================================================
========= -->
  <complexType name="PointPropertyType">
    <annotation>
      <documentation>
        Encapsulates a single point to represent position, location, or
        centerOf properties.
      </documentation>
    </annotation>
    <complexContent>
      <restriction base="gml:GeometryPropertyType">
        <sequence minOccurs="0">
          <element ref="gml:Point"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
      </restriction>
    </complexContent>
  </complexType>
  <complexType name="CurvePropertyType">
    <complexContent>
      <restriction base="gml:GeometryPropertyType">
        <sequence minOccurs="0">
          <element ref="gml:_Curve"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
      </restriction>
    </complexContent>
  </complexType>
  <complexType name="SurfacePropertyType">
    <complexContent>
      <restriction base="gml:GeometryPropertyType">
        <sequence minOccurs="0">
          <element ref="gml:_Surface"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
      </restriction>
    </complexContent>
  </complexType>
  <complexType name="LineStringPropertyType">
    <annotation>
      <documentation>
        Encapsulates a single LineString to represent centerLineOf or
        edgeOf properties.
      </documentation>
    </annotation>
    <complexContent>
      <restriction base="gml:CurvePropertyType">
        <sequence minOccurs="0">
          <element ref="gml:LineString"/>
        </sequence>
      </restriction>
    </complexContent>
  </complexType>
  <complexType name="PolygonPropertyType">
    <annotation>
      <documentation>
        Encapsulates a single polygon to represent coverage or extentOf
        properties.
      </documentation>
    </annotation>
    <complexContent>
```

```
    <restriction base="gml:SurfacePropertyType">
     <sequence minOccurs="0">
      <element ref="gml:Polygon"/>
     </sequence>
    </restriction>
   </complexContent>
  </complexType>
  <complexType name="MultiPointPropertyType">
   <annotation>
    <documentation>
     Encapsulates a MultiPoint element to represent the following
     discontiguous geometric properties: multiLocation, multiPosition,
     multiCenterOf.
    </documentation>
   </annotation>
   <complexContent>
    <restriction base="gml:GeometryPropertyType">
     <sequence minOccurs="0">
      <element ref="gml:MultiPoint"/>
     </sequence>
     <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </restriction>
   </complexContent>
  </complexType>
  <complexType name="MultiCurvePropertyType">
   <complexContent>
    <restriction base="gml:GeometryPropertyType">
     <sequence minOccurs="0">
      <element ref="gml:MultiCurve"/>
     </sequence>
    </restriction>
   </complexContent>
  </complexType>
  <complexType name="MultiSurfacePropertyType">
   <complexContent>
    <restriction base="gml:GeometryPropertyType">
     <sequence minOccurs="0">
      <element ref="gml:MultiSurface"/>
     </sequence>
    </restriction>
   </complexContent>
  </complexType>
  <complexType name="MultiLineStringPropertyType">
   <annotation>
    <documentation>
     Encapsulates a MultiLineString element to represent the following
     discontiguous geometric properties: multiEdgeOf, multiCenterLineOf.
    </documentation>
   </annotation>
   <complexContent>
    <restriction base="gml:MultiCurvePropertyType">
     <sequence minOccurs="0">
      <element ref="gml:MultiLineString"/>
     </sequence>
    </restriction>
   </complexContent>
  </complexType>
  <complexType name="MultiPolygonPropertyType">
   <annotation>
    <documentation>
     Encapsulates a MultiPolygon to represent the following discontiguous
     geometric properties: multiCoverage, multiExtentOf.
    </documentation>
   </annotation>
   <complexContent>
    <restriction base="gml:MultiSurfacePropertyType">
     <sequence minOccurs="0">
      <element ref="gml:MultiPolygon"/>
     </sequence>
    </restriction>
   </complexContent>
  </complexType>
  <complexType name="MultiGeometryPropertyType">
   <annotation>
    <documentation>Encapsulates a MultiGeometry
element.</documentation>
   </annotation>
   <complexContent>
    <restriction base="gml:GeometryPropertyType">
     <sequence minOccurs="0">
      <element ref="gml:MultiGeometry"/>
     </sequence>
     <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </restriction>
   </complexContent>
  </complexType>
  <complexType name="ComplexPropertyType">
   <complexContent>
    <restriction base="gml:GeometryPropertyType">
     <sequence minOccurs="0">
      <element ref="gml:GeometryComplex"/>
     </sequence>
    </restriction>
   </complexContent>
  </complexType>
  <simpleType name="NullType">
   <annotation>
    <documentation>
     If a bounding shape is not provided for a feature collection,
     explain why. Allowable values are:
     innapplicable - the features do not have geometry
     unknown - the boundingBox cannot be computed
     unavailable - there may be a boundingBox but it is not divulged
     missing - there are no features
    </documentation>
   </annotation>
   <restriction base="string">
    <enumeration value="inapplicable"/>
    <enumeration value="unknown"/>
    <enumeration value="unavailable"/>
    <enumeration value="missing"/>
   </restriction>
  </simpleType>
</schema>
```

Appendix D Source code

D.1 Scripts

D.1.1 arc2ora.sh

```
#!/bin/sh
#
#  creation date: 07 05 2001
#        author: FM
#   dependencies: unloadarc.sh loadora_geom.pl loadora_topo.pl
#                 wings.pl join.pl
#         usage: arc2ora.sh <coverage> <database> [<extent>]
#    description: creates a oracle database from arcinfo data from arcinfo data.
#

if test -z "$2"
then
   echo "\nusage: $0 <coverage> <database> [<extent>]\n"
   exit
fi

set -x

unloadarc.sh $1

cat ${1}.edges | loadora_topo.pl -a $2 ${1}_curve_admin
cat ${1}.nodes | loadora_topo.pl -n $2 ${1}_point_admin
cat ${1}.faces | loadora_topo.pl -p $2 ${1}_surface_admin

cat ${1}.lines | loadora_geom.pl -A $2 ${1}_curve_geom
cat ${1}.points | loadora_geom.pl -n ${2} ${1}_point_geom
cat ${1}.polygons | loadora_geom.pl -P ${2} ${1}_surface_geom

wings.pl -f $2 $1

rm -f ${1}.edges
rm -f ${1}.nodes
rm -f ${1}.faces
rm -f ${1}.lines
rm -f ${1}.points
rm -f ${1}.polygons
```

D.1.2 unloadarc.sh

```
#!/bin/sh

# usage: unloadarc.sh <coverage>

set -x

rm -f ${1}.nodes
rm -f ${1}.edges
rm -f ${1}.faces
rm -f ${1}.points
rm -f ${1}.lines
rm -f ${1}.polygons


arc << EOF

tables
select ${1}.aat
calculate ${1}-ID = ${1}#
unload ${1}.edges FNODE#,TNODE#,LPOLY#,RPOLY#,LENGTH,${1}#,${1}-
ID,IGDS-GGNO,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,SYMBOL init
quit
idedit ${1} line
ungenerate line ${1} ${1}.lines

tables
select ${1}.pat
calculate ${1}-ID = ${1}#
unload ${1}.faces AREA,PERIMETER,${1}#,${1}-ID,IGDS-
GGNO,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,SYMBOL init
quit
idedit ${1} poly
ungenerate poly ${1} ${1}.polygons

tables
select ${1}.nat
calculate ${1}-ID = ${1}#
unload ${1}.nodes ARC#,${1}#,${1}-ID init
quit
idedit ${1} point
nodepoint ${1} ${1}NP
build ${1}NP point
ungenerate point ${1}NP ${1}.points

quit
EOF
```

D.1.3 loadora_geom.pl

```
#!/usr/local/bin/perl
#
#  creation date: 07 05 2001
#        author: FM
#       based on: arc2ora.perl (by W.Quak)
#   dependencies: insert_metadata.sh (aka insert_into_user_sdo_geom_metadata.sql
by W.Quak)
#         usage: loadora_geom.pl <-a|-n|-p> <database> <geometryfile>
#    description: loads geometry ascii file created with arc into oracle.
#
#
if ($#ARGV < 2)
{
   print STDERR "\nusage: $0 <-a|-n|-p> <database> <geometryfile>\n\n";
   exit
}

sub printpolystringoracle
{
   @temp = split(' ',$_[0]);
   #shift(@temp);
   $aantp = $_[1] ;

   print "2002,1,2,1|";

   ($x,$y) = split(',',shift(@temp));
   print "$x,$y";
   foreach(@temp)
   {
      ($x,$y) = split(',',$_);
      print ",$x,$y";
   }
   print "|";

   if ($option eq "-A") {
      print $aantp;
   }
}

sub printpolygonoracle
{
   @temp = split(' ',$_[1]);
   @temp2 = split (' ',$_[0]);
   @temp3 = split(' ',$_[1]);
   @temp4 = splice(@temp3,0,$temp2[0]);
   $box = $_[2] . "," . $_[3] . "," . $_[4] . "," . $_[5] ;
   pop(@temp2);
   #shift(@temp);

   if ($option eq "-P") {
      print "2003,1,1003,3|";
      print $box;
      print "|";
   }

   print "2003,1,1003,1|";
   ($x,$y) = split(',',shift(@temp4));
   print "$x,$y";
   foreach(@temp4)
   {
      ($x,$y) = split(',',$_);
      print ",$x,$y";
   }
   print "|";

   print "2003,1,1003,1";
   $offset = 1;
   foreach(@temp2)
   {
      $offset += $_*2;
      print "," . $offset . ",2003,1";
   }
   print "|";
   ($x,$y) = split(',',shift(@temp));
   print "$x,$y";
   foreach(@temp)
   {
      ($x,$y) = split(',',$_);
      print ",$x,$y";
   }
   print "|";
}

sub printpointoracle
{
   @temp = split(' ',$_[0]);

   print "2001,";

   ($x,$y) = split(',',shift(@temp));
   print "$x,$y";
}

#
# Get the name of the database table from the command line.
#
$option = shift;
$DATABASE = shift;
$tablename = uc(shift);

#
# Create the sql file which create the table.
#
open(MODEL,"> $tablename.sql");
if ($option eq "-P")
{
   print MODEL "set echo on;\n";
   print MODEL "\n";
   print MODEL "drop table $tablename;\n";
   print MODEL "create table $tablename\n";
   print MODEL "(\n";
   print MODEL "   oid number(11) not null,\n";
   print MODEL "   box mdsys.sdo_geometry,\n";
   print MODEL "   convex mdsys.sdo_geometry,\n";
   print MODEL "   geometry mdsys.sdo_geometry\n";
   print MODEL ");\n";
   print MODEL "\n";
   print MODEL "exit;\n";
}
elsif ($option eq "-A")
{
   print MODEL "set echo on;\n";
   print MODEL "\n";
   print MODEL "drop table $tablename;\n";
   print MODEL "create table $tablename\n";
   print MODEL "(\n";
   print MODEL "   oid number(11) not null,\n";
   print MODEL "   geometry mdsys.sdo_geometry,\n";
   print MODEL "   pcount number(11)\n";
   print MODEL ");\n";
   print MODEL "\n";
   print MODEL "exit;\n";
}
else
{
   print MODEL "set echo on;\n";
   print MODEL "\n";
   print MODEL "drop table $tablename;\n";
```

```
  print MODEL "create table $tablename\n";
  print MODEL "(\n";
  print MODEL "   oid number(11) not null,\n";
  print MODEL "   geometry mdsys.sdo_geometry\n";
  print MODEL ");\n";
  print MODEL "\n";
  print MODEL "exit;\n";
}
close(MODEL);

#
# Create the controlfile
#
open(CONTROL,"> $tablename.ctl");
if ($option eq "-P")
{
  print CONTROL "load data\n";
  print CONTROL "infile '$tablename.dat' \"str\"\n";
  print CONTROL "into table $tablename\n";
  print CONTROL "fields terminated by ',\n";
  print CONTROL "trailing nullcols (\n";
  print CONTROL "   oid integer external,\n";
  print CONTROL "   box column object\n";
  print CONTROL "   (\n";
  print CONTROL "      sdo_gtype integer external,\n";
  print CONTROL "      sdo_elem_info varray terminated by '|'\n";
  print CONTROL "         (till_element_info integer external),\n";
  print CONTROL "      sdo_ordinates varray terminated by ')'\n";
  print CONTROL "         (after_element_info float external)\n";
  print CONTROL "   ),\n";
  print CONTROL "   convex column object\n";
  print CONTROL "   (\n";
  print CONTROL "      sdo_gtype integer external,\n";
  print CONTROL "      sdo_elem_info varray terminated by '|'\n";
  print CONTROL "         (till_element_info integer external),\n";
  print CONTROL "      sdo_ordinates varray terminated by ')'\n";
  print CONTROL "         (after_element_info float external)\n";
  print CONTROL "   ),\n";
  print CONTROL "   geometry column object\n";
  print CONTROL "   (\n";
  print CONTROL "      sdo_gtype integer external,\n";
  print CONTROL "      sdo_elem_info varray terminated by '|'\n";
  print CONTROL "         (till_element_info integer external),\n";
  print CONTROL "      sdo_ordinates varray terminated by ')'\n";
  print CONTROL "         (after_element_info float external)\n";
  print CONTROL "   )\n";
  print CONTROL ")\n";
}
elsif ($option eq "-n")
{
  print CONTROL "load data\n";
  print CONTROL "infile '$tablename.dat' \"str\"\n";
  print CONTROL "into table $tablename\n";
  print CONTROL "fields terminated by ',\n";
  print CONTROL "trailing nullcols (\n";
  print CONTROL "   oid integer external,\n";
  print CONTROL "   geometry column object\n";
  print CONTROL "   (\n";
  print CONTROL "      sdo_gtype integer external,\n";
  print CONTROL "      sdo_point column object\n";
  print CONTROL "         (x integer external,\n";
  print CONTROL "          y integer external)\n";
  print CONTROL "   )";
  print CONTROL ")\n";
}
elsif ($option eq "-A")
{
  print CONTROL "load data\n";
  print CONTROL "infile '$tablename.dat' \"str\"\n";
  print CONTROL "into table $tablename\n";
  print CONTROL "fields terminated by ',\n";
  print CONTROL "trailing nullcols (\n";
  print CONTROL "   oid integer external,\n";
  print CONTROL "   geometry column object\n";
  print CONTROL "   (\n";
  print CONTROL "      sdo_gtype integer external,\n";
  print CONTROL "      sdo_elem_info varray terminated by '|'\n";
  print CONTROL "         (till_element_info integer external),\n";
  print CONTROL "      sdo_ordinates varray terminated by ')'\n";
  print CONTROL "         (after_element_info float external)\n";
  print CONTROL "   ),\n";
  print CONTROL "   pcount integer external\n";
  print CONTROL ")\n";
}
else
{
  print CONTROL "load data\n";
  print CONTROL "infile '$tablename.dat' \"str\"\n";
  print CONTROL "into table $tablename\n";
  print CONTROL "fields terminated by ',\n";
  print CONTROL "trailing nullcols (\n";
  print CONTROL "   oid integer external,\n";
  print CONTROL "   geometry column object\n";
  print CONTROL "   (\n";
  print CONTROL "      sdo_gtype integer external,\n";
  print CONTROL "      sdo_elem_info varray terminated by '|'\n";
  print CONTROL "         (till_element_info integer external),\n";
  print CONTROL "      sdo_ordinates varray terminated by ')'\n";
  print CONTROL "         (after_element_info float external)\n";
  print CONTROL "   )\n";
  print CONTROL ")\n";
}
close(CONTROL);

#
# Oracle spatial does not like long index names.
#
$INDEXNAME = substr($tablename,0,16) . "_1";
$INDEXNAME2 = substr($tablename,0,16) . "_2";
$INDEXNAME3 = substr($tablename,0,16) . "_3";

open(INDEX,"> $tablename-index.sql");
print INDEX "set echo on;\n";
print INDEX "\n";
print INDEX "drop index ${INDEXNAME} force;\n";
print INDEX "\n";
print INDEX "create index ${INDEXNAME} on $tablename(geometry)\n";
print INDEX "indextype is mdsys.spatial_index\n";
print INDEX "parameters ('initial=1m next=1m pctincrease=0 sdo_fanout=32');\n";
print INDEX "\n";
print INDEX "analyze table ${INDEXNAME}_rt\$ compute statistics;\n";
print INDEX "analyze table $tablename compute statistics;\n";
print INDEX "\n";
if ($option eq "-P")
{
  print INDEX "drop index ${INDEXNAME2} force;\n";
  print INDEX "\n";
  print INDEX "create index ${INDEXNAME2} on $tablename(box)\n";
  print INDEX "indextype is mdsys.spatial_index\n";
  print INDEX "parameters ('initial=1m next=1m pctincrease=0 sdo_fanout=32');\n";
  print INDEX "\n";
  print INDEX "analyze table ${INDEXNAME2}_rt\$ compute statistics;\n";
  print INDEX "analyze table $tablename compute statistics;\n";
  print INDEX "\n";

  print INDEX "drop index ${INDEXNAME3} force;\n";
  print INDEX "\n";
  print INDEX "create index ${INDEXNAME3} on $tablename(convex)\n";
  print INDEX "indextype is mdsys.spatial_index\n";
  print INDEX "parameters ('initial=1m next=1m pctincrease=0 sdo_fanout=32');\n";
  print INDEX "\n";
  print INDEX "analyze table ${INDEXNAME3}_rt\$ compute statistics;\n";
  print INDEX "analyze table $tablename compute statistics;\n";
  print INDEX "\n";
}
print INDEX "GRANT SELECT on $tablename to public;\n";
print INDEX "exit;\n";
close(INDEX);

#
# Create the datafile
#
open(DATA,"> $tablename.dat");
select(DATA);


$extminx =  100000000000 ;
$extmaxx = -100000000000 ;
$extminy =  100000000000 ;
$extmaxy = -100000000000 ;
if (($option eq "-a") || ($option eq "-A"))
{
  while(<>)
  {
    chop;
    if (/END/)
    {
      last;
    }
    $oid = int($_);
    $polystring = "";
    while(<>)
    {
      if(/END/)
      {
        $endx = $x;
        $endy = $y;
      last;
      }
      else
      {
      ($x,$y) = split(' ');
        if ($polystring eq "")
        {
          $startx = $x;
          $starty = $y;
          $aantpunten = 0;
        }
      $polystring .= $x . "," . $y . " " ;
        $aantpunten++ ;
        if ($x<$extminx) { $extminx = $x; }
        if ($x>$extmaxx) { $extmaxx = $x; }
        if ($y<$extminy) { $extminy = $y; }
        if ($y>$extmaxy) { $extmaxy = $y; }
      }
    }
    chop($polystring);
    print "$oid,";
    &printpolystringoracle($polystring,$aantpunten);
    print "\n";
  }
}
elsif ($option eq "-n")
{
  while(<>)
  {
    chop;
    ($n,$x,$y) = split(' ');
    $polystring = $x . "," . $y ;
    $oid = int($n);
    print "$oid,";
    &printpointoracle($polystring);
    print "\n";
    if ($n ne "END")
    {
      if ($x<$extminx) { $extminx = $x; }
      if ($x>$extmaxx) { $extmaxx = $x; }
      if ($y<$extminy) { $extminy = $y; }
      if ($y>$extmaxy) { $extmaxy = $y; }
    }
  }
}
elsif (($option eq "-p") || ($option eq "-P"))
{
  $polystring = "";
  $info = "";
  while(<>)
  {
    chop;
    if (/END/)
    {
      if ($polystring ne "")
      {
        chop($polystring);
        print "$oid,";
        &printpolygonoracle($info,$polystring,$minx,$miny,$maxx,$maxy);
        print "\n";
        $polystring = "";
        $info ="";
      }
      last;
    }
    ($n,$x,$y) = split(' ');
    $n = int($n);
    if($n>=0)
    {
      if ($polystring ne "")
      {
```

```
            chop($polystring);
            print "$oid,";
            &printpolygonoracle($info,$polystring,$minx,$miny,$maxx,$maxy);
            print "\n";
            $polystring = "";
            $info ="";
          }
          $oid = $n;
          $minx = $x;
          $maxx = $x;
          $miny = $y;
          $maxy = $y;
        }
        $ring = "";
        $coordcount = 0;
        while(<>)
        {
          if(/END/)
          {
          last;
          }
          else
          {
          ($x,$y) = split(' ');
          $ring .= $x . "," . $y . " " ;
            $coordcount += 1;
            if ($x<$minx) { $minx = $x; }
            if ($x>$maxx) { $maxx = $x; }
            if ($y<$miny) { $miny = $y; }
            if ($y>$maxy) { $maxy = $y; }
            if ($x<$extminx) { $extminx = $x; }
            if ($x>$extmaxx) { $extmaxx = $x; }
            if ($y<$extminy) { $extminy = $y; }
            if ($y>$extmaxy) { $extmaxy = $y; }
          }
        }
        chop($ring);
        $polystring .= $ring . " " ;
        $info .= $coordcount . " " ;
      }
}
close(DATA);
select(STDOUT);

$extent = $extminx . " " . $extmaxx . " " . $extminy . " " . $extmaxy ;
$precision = 0.5 ;
print `sqlplus $DATABASE \@$tablename.sql`;
print `sqlldr $DATABASE CONTROL=$tablename.ctl`;
print `insert_metadata.sh $DATABASE $tablename GEOMETRY $extent $precision`;
if ($option eq "-P")
{
   print `insert_metadata.sh $DATABASE $tablename BOX $extent $precision`;
   print `insert_metadata.sh $DATABASE $tablename CONVEX $extent $precision`;
}
print `sqlplus $DATABASE \@$tablename-index`;

print `rm -f $tablename.ctl`;
print `rm -f $tablename.log`;
print `rm -f $tablename.dat`;
print `rm -f $tablename.sql`;
print `rm -f ${tablename}-index.sql`;
```

D.1.4 loadora_topo.pl

```
#!/usr/local/bin/perl
#
#  creation date: 07 05 2001
#        author: FM
#       based on:
#   dependencies: lki_ing2ora.pl (by W.Quak)
#          usage: topo_arc2ora.pl <-a|-n|-p> <database> <tablename> [<extent>]
#    description: loads topology ascii file created with arc into oracle.
#


if ($#ARGV < 2)
{
   print STDERR "\nusage: $0 <-a|-n|-p> <database> <tablename> [<extent>]\n\n";
   exit
}

$option = shift;
$DATABASE = shift;
$tablename = uc(shift);
$extent = $_ ;

#
#
#
$datamodel{-a} =
  "    FNODE_ID number(11),\n" .
  "    TNODE_ID number(11),\n" .
  "    LPOLY_ID number(11),\n" .
  "    RPOLY_ID number(11),\n" .
  "    LENGTH number(11),\n" .
  "    ID number(11),\n" .
  "    EXTID number(11),\n" .
  "    IGDS_GG number(11),\n" .
  "    C1 number(11),\n" .
  "    C2 number(11),\n" .
  "    C3 number(11),\n" .
  "    C4 number(11),\n" .
  "    C5 number(11),\n" .
  "    C6 number(11),\n" .
  "    C7 number(11),\n" .
  "    C8 number(11),\n" .
  "    C9 number(11),\n" .
  "    C10 number(11),\n" .
  "    SYMBOL number(11)\n";

$attrlist{-a} =
       "-integer FNODE_ID " .
       "-integer TNODE_ID " .
       "-integer LPOLY_ID " .
       "-integer RPOLY_ID " .
       "-integer LENGTH " .
       "-integer ID " .
       "-integer EXTID " .
       "-integer IGDS_GG " .
       "-integer C1 " .
       "-integer C2 " .
       "-integer C3 " .
       "-integer C4 " .
       "-integer C5 " .
       "-integer C6 " .
       "-integer C7 " .
       "-integer C8 " .
       "-integer C9 " .
       "-integer C10 " .
       "-integer SYMBOL ";

$datamodel{-n} =
       "    ARC_ID number(11),\n" .
       "    ID number(11),\n" .
       "    EXTID number(11)\n";

$attrlist{-n} =
       "-integer ARC_ID " .
       "-integer ID " .
       "-integer EXTID ";

$datamodel{-p} =
       "    AREA number(11),\n" .
       "    PERIMETER number(11),\n" .
       "    ID number(11),\n" .
       "    EXTID number(11),\n" .
       "    IGDS_GG number(11),\n" .
       "    C1 number(11),\n" .
       "    C2 number(11),\n" .
       "    C3 number(11),\n" .
       "    C4 number(11),\n" .
       "    C5 number(11),\n" .
       "    C6 number(11),\n" .
       "    C7 number(11),\n" .
       "    C8 number(11),\n" .
       "    C9 number(11),\n" .
       "    C10 number(11),\n" .
       "    SYMBOL number(11)\n";

$attrlist{-p} =
       "-integer AREA " .
       "-integer PERIMETER " .
       "-integer ID " .
       "-integer EXTID " .
       "-integer IGDS_GG " .
       "-integer C1 " .
       "-integer C2 " .
       "-integer C3 " .
       "-integer C4 " .
       "-integer C5 " .
       "-integer C6 " .
       "-integer C7 " .
       "-integer C8 " .
       "-integer C9 " .
       "-integer C10 " .
       "-integer SYMBOL ";

#
#
#
open(MODEL,"> $tablename.sql");
print MODEL "\n";
print MODEL "drop table $tablename;\n";
print MODEL "create table $tablename\n";
print MODEL "(\n";
print MODEL "$datamodel{$option}";
print MODEL ");\n";
print MODEL "\n";
print MODEL "quit;\n";
close(MODEL);


print `sqlplus $DATABASE \@$tablename.sql`;
print `sed 's/,/    /g' | lki_ing2ora.pl all $tablename $attrlist{$option} > $tablename.dat`;
print `sqlldr $DATABASE control=$tablename.ctl  data=$tablename.dat`;


print `rm -f $tablename.sql`;
print `rm -f $tablename.ctl`;
print `rm -f $tablename.dat`;
print `rm -f $tablename.log`;
```

D.1.5 wings.pl

```
#!/usr/local/bin/perl
#
#  creation date: 07 05 2001
#        author: FM
#       based on:
#   dependencies:
#          usage: wings.pl <database> <pointadmintable> <curveadmintable>
<surfaceadmintable> <pointgeomtable> <curvegeomtable> <surfacegeomtable>
#    description: creates database table with winged edge information.
#

if ($ARGV[0] eq "-f")
{
   shift;
   $DATABASE = shift;
   $base = uc(shift);
   $pointtable = $base . "_POINT_ADMIN";
   $curvetable = $base . "_CURVE_ADMIN";
   $surfacetable = $base . "_SURFACE_ADMIN";
   $pointgtable = $base . "_POINT_GEOM";
   $curvegtable = $base . "_CURVE_GEOM";
   $surfacegtable = $base . "_SURFACE_GEOM";
}
else
{
   if($#ARGV < 6)
   {
      print STDERR "\nusage: $0 <database> <pointadmintable> <curveadmintable>
<surfaceadmintable> <pointgeomtable> <curvegeomtable> <surfacegeomtable>\n\n";
      exit
   }
   $DATABASE = shift;
   $pointtable = uc(shift);
   $curvetable = uc(shift);
   $surfacetable = uc(shift);
   $pointgtable = uc(shift);
   $curvegtable = uc(shift);
   $surfacegtable = uc(shift);
   $pointtable =~ /([^_])+_/;
   $base = $1;
```

```
}

$sqlfile = "wings.sql";

#
#
#
open(MODEL,"> $sqlfile");
print MODEL qq~
-- $sqlfile
-- generated by $0
--

drop table bridgearc;
drop table d0arc;
drop table d0node;
drop table d1node;
drop table d2node;
drop table isoarc;
drop table isonode;
drop table isopoly;
drop table nawbase;
drop table ncwbase;
drop table pawbase;
drop table pcwbase;
drop table naw;
drop table ncw;
drop table paw;
drop table pcw;
drop table wings;
drop table narcbase;
drop table parcbase;
drop table narc;
drop table parc;
drop table node;
drop table edge;
drop table netnode;
drop table netedge;
drop table innerring;
drop table outerring;
drop table ${base}_node;
drop table ${base}_edge;
drop table ${base}_face;
drop table ${base}_ring;
drop table ${base}_universe;
drop table ${base}_ffeature;
drop table ${base}_efeature;
drop view ${base}_ffeature2;
drop view ${base}_efeature2;


create table curve
as
  select *
  from $curvegtable
;

update curve set geometry = NULL where pcount = 2;



create table edge
as
  select id,fnode_id,tnode_id,lpoly_id,rpoly_id
  from   $curvetable
  where  lpoly_id != rpoly_id
;


create table node
as
select node_id as id, count(arc_id) as d from (
select id as arc_id, fnode_id as node_id, 'f' from edge
union
select id as arc_id, tnode_id as node_id, 't' from edge
)
group by node_id
;

create table d2node
as
select id
from   node
where  d=2
;

create table d1node
as
select id
from   node
where  d=1
;

create table bridgearc
as
select id, fnode_id, tnode_id, lpoly_id as poly_id
from   $curvetable
where  lpoly_id = rpoly_id
and    (fnode_id in (select id from node)
       and tnode_id in (select id from node))
;

create table isoarc
as
select id, fnode_id, tnode_id, lpoly_id as poly_id
from   $curvetable
where  lpoly_id = rpoly_id
minus
select *
from bridgearc
;

create table netnode
as
select node_id as id, count(arc_id) as d from (
select id as arc_id, fnode_id as node_id, 'f' from isoarc
union
select id as arc_id, tnode_id as node_id, 't' from isoarc
)
group by node_id
;

create table netedge
as
select *
from isoarc
where fnode_id in (select id from netnode where d>1)
or    tnode_id in (select id from netnode where d>1)
;

delete from isoarc
where id in (select id from netedge)
;

create table d0node
as
select id
from   $pointtable
minus
select id
from   node
minus
select id
from   (select id from netnode)
;
create table isonode
as
select n.id, p.oid as poly_id
from   d0node n, $pointgtable p, $surfacegtable s
where  sdo_relate(p.geometry,s.geometry,'mask=INSIDE querytype=JOIN') = 'TRUE'
;



create table ncwbase
as
select a.id, a2.id as ncw, a.lpoly_id, a.rpoly_id, a2.lpoly_id as npoly_id
from   edge a, edge a2
where  a.tnode_id = a2.fnode_id and a.rpoly_id = a2.rpoly_id and a.id != a2.id
union
select a.id, a2.id as ncw, a.lpoly_id, a.rpoly_id, a2.rpoly_id as npoly_id
from   edge a, edge a2
where  a.tnode_id = a2.tnode_id and a.rpoly_id = a2.lpoly_id and a.id != a2.id
union
select a.id, a.id as ncw, a.lpoly_id, a.rpoly_id, a.lpoly_id as npoly_id
from   edge a
where  a.fnode_id = a.tnode_id
;

create table nawbase
as
select a.id, a2.id as naw, a.lpoly_id, a.rpoly_id, a2.rpoly_id as npoly_id
from   edge a, edge a2
where  a.fnode_id = a2.tnode_id and a.lpoly_id = a2.lpoly_id and a.id != a2.id
union
select a.id, a2.id as naw, a.lpoly_id, a.rpoly_id, a2.lpoly_id as npoly_id
from   edge a, edge a2
where  a.fnode_id = a2.fnode_id and a.lpoly_id = a2.rpoly_id and a.id != a2.id
union
select a.id, a.id as naw, a.lpoly_id, a.rpoly_id, a.rpoly_id as npoly_id
from   edge a
where  a.fnode_id = a.tnode_id
;

create table pcwbase
as
select a.id, a2.id as pcw, a.lpoly_id, a.rpoly_id, a2.lpoly_id as npoly_id
from   edge a, edge a2
where  a.fnode_id = a2.tnode_id and a.rpoly_id = a2.rpoly_id and a.id != a2.id
union
select a.id, a2.id as pcw, a.lpoly_id, a.rpoly_id, a2.rpoly_id as npoly_id
from   edge a, edge a2
where  a.fnode_id = a2.fnode_id and a.rpoly_id = a2.lpoly_id and a.id != a2.id
union
select a.id, a.id as pcw, a.lpoly_id, a.rpoly_id, a.lpoly_id as npoly_id
from   edge a
where  a.fnode_id = a.tnode_id
;

create table pawbase
as
select a.id, a2.id as paw, a.lpoly_id, a.rpoly_id, a2.rpoly_id as npoly_id
from   edge a, edge a2
where  a.tnode_id = a2.fnode_id and a.lpoly_id = a2.lpoly_id and a.id != a2.id
union
select a.id, a2.id as paw, a.lpoly_id, a.rpoly_id, a2.lpoly_id as npoly_id
from   edge a, edge a2
where  a.tnode_id = a2.tnode_id and a.lpoly_id = a2.rpoly_id and a.id != a2.id
union
select a.id, a.id as paw, a.lpoly_id, a.rpoly_id, a.rpoly_id as npoly_id
from   edge a
where  a.fnode_id = a.tnode_id
;

create table ncw
as
select * from ncwbase
minus
select * from ncwbase
where  lpoly_id = npoly_id
and    id in (select id from ncwbase group by id having count(ncw) > 1)
;

create table naw
as
select * from nawbase
minus
select * from nawbase
where  rpoly_id = npoly_id
and    id in (select id from nawbase group by id having count(naw) > 1)
;

create table pcw
as
select * from pcwbase
minus
select * from pcwbase
where  lpoly_id = npoly_id
and    id in (select id from pcwbase group by id having count(pcw) > 1)
;

create table paw
as
select * from pawbase
minus
select * from pawbase
where  rpoly_id = npoly_id
and    id in (select id from pawbase group by id having count(paw) > 1)
```

```
;


create table parcbase
as
select a.id, a2.id as parc_id, a.poly_id, a2.lpoly_id as ppoly_id
from   bridgearc a, edge a2
where  a.fnode_id = a2.tnode_id and a.poly_id = a2.rpoly_id
union
select a.id, a2.id as parc_id, a.poly_id, a2.rpoly_id as ppoly_id
from   bridgearc a, edge a2
where  a.fnode_id = a2.fnode_id and a.poly_id = a2.lpoly_id
;

create table narcbase
as
select a.id, a2.id as narc_id, a.poly_id, a2.lpoly_id as npoly_id
from   bridgearc a, edge a2
where  a.tnode_id = a2.fnode_id and a.poly_id = a2.rpoly_id
and    a2.lpoly_id not in (select ppoly_id from parcbase)
union
select a.id, a2.id as narc_id, a.poly_id, a2.rpoly_id as npoly_id
from   bridgearc a, edge a2
where  a.tnode_id = a2.tnode_id and a.poly_id = a2.lpoly_id
and    a2.rpoly_id not in (select ppoly_id from parcbase)
;

create table parc
as
select * from parcbase
where  parc_id in (select min(parc_id)
                   from   parcbase
                   group by poly_id, ppoly_id)
;

create table narc
as
select * from narcbase
where narc_id in (select min(narc_id)
                  from   narcbase
                  group by poly_id, npoly_id)
;

create table innerring
as
select a.poly_id, a.narc_id as sarc_id
from   narc a, $surfacegtable p, $surfacegtable p2
where  a.poly_id = p.oid and a.npoly_id = p2.oid
and    sdo_relate(p.convex,p2.convex,'mask=contains querytype=join') = 'TRUE'
union
select a.poly_id, a.parc_id as sarc_id
from   parc a, $surfacegtable p, $surfacegtable p2
where  a.poly_id = p.oid and a.ppoly_id = p2.oid
and    sdo_relate(p.convex,p2.convex,'mask=contains querytype=join') = 'TRUE'
;

create table outerring
as
select poly_id, min(id) as sarc_id
from   (select a.poly_id, a.narc_id as id
       from   narc a, $surfacegtable p, $surfacegtable p2
       where  a.poly_id = p.oid and a.npoly_id = p2.oid
       and    sdo_relate(p.convex,p2.convex,'mask=contains querytype=join') =
'FALSE'
       union
       select a.poly_id, a.parc_id as id
       from   parc a, $surfacegtable p, $surfacegtable p2
       where  a.poly_id = p.oid and a.ppoly_id = p2.oid
       and    sdo_relate(p.convex,p2.convex,'mask=contains querytype=join') =
'FALSE'
       union
       select lpoly_id as poly_id, id
       from   edge
       where  lpoly_id not in (select poly_id from innerring)
       union
       select rpoly_id as poly_id, id
       from   edge
       where  rpoly_id not in (select poly_id from innerring))
group by poly_id
;

create table wings
as
select edge.id, ncw.ncw, pcw.pcw, naw.naw, paw.paw
from   edge, ncw, pcw, naw, paw
where  edge.id = ncw.id
and    edge.id = pcw.id
and    edge.id = naw.id
and    edge.id = paw.id
;



create table ${base}_node
as
select n.id, g.geometry
from   node n, $pointgtable g
where  n.id = g.oid
;

create table ${base}_edge
as
select e.id, e.fnode_id as startNode, e.tnode_id as endNode,
      e.lpoly_id as leftFace, e.rpoly_id as rightFace,
      w.naw as nawEdge, w.ncw as ncwEdge, g.geometry
from   edge e, wings w, $curvegtable g
where  e.id = g.oid
and    e.id = w.id
;

create table ${base}_face
as
select poly_id as id, sarc_id as startEdge
from   outerring
;

create table ${base}_ring
as
select poly_id as id, sarc_id as startEdge
from   innerring
;

create table ${base}_ffeature
as
select extid as fid, c1 as tdncode, area, id as surfaceProperty
from  $surfacetable
;

create table ${base}_efeature
as
select a.extid as fid, a.c1 as tdncode, a.length, g.geometry as curveProperty
from $curvetable a, $curvegtable g
where a.extid = g.oid
and a.id in (select id from isoarc
             union
             select id from netedge)
;

create view ${base}_ffeature2
as
select f.fid, f.tdncode, f.area, g.geometry as surfaceProperty
from   ${base}_ffeature f, $surfacegtable g
where  f.fid = g.oid
;

create view ${base}_efeature2
as
select f.fid, f.tdncode, f.length, g.geometry as curveProperty
from   ${base}_efeature f, $curvegtable g
where  f.fid = g.oid
;


create table ${base}_universe
as
select *
from   ${base}_face
where  id=1
;

delete from ${base}_face where id=1;
update ${base}_edge set leftFace=NULL , nawEdge=NULL where leftFace=1;
update ${base}_edge set rightFace=NULL , ncwEdge=NULL where rightFace=1;
delete from ${base}_ffeature where surfaceProperty=1;


-- drop table bridgearc;
-- drop table d0arc;
-- drop table d0node;
-- drop table d1node;
-- drop table d2node;
-- drop table isoarc;
-- drop table isonode;
-- drop table isopoly;
-- drop table nawbase;
-- drop table ncwbase;
-- drop table pawbase;
-- drop table pcwbase;
-- drop table naw;
-- drop table ncw;
-- drop table paw;
-- drop table pcw;
-- drop table wings;
-- drop table narcbase;
-- drop table parcbase;
-- drop table narc;
-- drop table parc;
-- drop table node;
-- drop table edge;
-- drop table netnode;
-- drop table netedge;
-- drop table innerring;
-- drop table outerring;
-- drop table complex_curve;
-- drop table complex_innerring;
-- drop table complex_point;
-- drop table complex_surface;
-- drop table curve_feature;
-- drop table surface_feature;


exit;
~;
close(MODEL);

print `sqlplus $DATABASE \@$sqlfile`;

print `rm -f $sqlfile`;
```

D.2 GML

D.2.1 SDOtoMYGML.java

```
import java.io.*;
import java.sql.*;
import java.util.*;
import oracle.sql.STRUCT;
import oracle.jdbc.driver.*;
import oracle.sdoapi.OraSpatialManager;
import oracle.sdoapi.adapter.*;
import oracle.sdoapi.geom.*;



/**
 * This sample program reads all geometries from a database table and
 * converts them to the GML format.
 */
public class SDOtoMYGML
{
  // static data members
  static String complexBase, featureTable, outputBaseName;
  static String uPrefix = new String("tdn:");
  static String gPrefix = new String("gml:");
  static FileOutputStream oS, oS2;
  static PrintStream pS, xsd;
  static GeometryAdapter gmlAdapter;
  static GeometryAdapter sdoAdapter;

  //
  // Convert SDO objects to GML
  //
  public static void main(String args[]) throws Exception
```

```
  {
    if (args.length < 2)
    {
      System.out.println("Parameters:");
      System.out.println("<Complex Base name>: base name of the complex tables");
      System.out.println("<Feature Table name>: Table name of an input table");
      return;
    }

    outputBaseName = args[0];
    complexBase = args[0];
    featureTable = args[1];


    //
    // Connect to the database
    //
    Connection conn =

DriverManager.getConnection("jdbc:oracle:thin:@www.gdmc.nl:1521:igis","franklin",
"franklin");
    ((OracleConnection)conn).setDefaultRowPrefetch(100);

    // First have to register GML geometry adapter with the OraSpatialManager
    // because this adapter is not automatically registered
    // (in contrast to SDO, WKB, and WKT adapters)
    if (!OraSpatialManager.registerGeometryAdapter(
          new AdapterGML(OraSpatialManager.getGeometryFactory())))
    {
      System.out.println("The GML geometry adapter could not be registered.");
      return;
    }

    System.out.println("query");
    // Query for SDO_GEOMETRY object
    xsd = new PrintStream(new FileOutputStream(outputBaseName + ".xsd"));
    pS = new PrintStream(new FileOutputStream(outputBaseName + ".gml"));

    //
    // Write Stylesheet header.
    //
    xsd.println("<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\"?>");
    xsd.println("<!-- File: " + outputBaseName + ".xsd  -->");
    xsd.println("<schema");
    xsd.println(" targetNamespace=\"http://www.tdn.nl/top10test\"");
    xsd.println(" xmlns=\"http://www.w3.org/2000/10/XMLSchema\"");
    xsd.println(" xmlns:xlink=\"http://www.w3.org/1999/xlink\"");
    xsd.println(" xmlns:gml=\"http://www.opengis.net/gml\"");
    xsd.println(" xmlns:tdn=\"http://www.tdn.nl/top10test\"");
    xsd.println(" elementFormDefault=\"qualified\"");
    xsd.println(" version=\"0.4\">\n");

    xsd.println("\t<import namespace=\"http://www.opengis.net/gml\"
schemaLocation=\"feature.xsd\"/>");
    xsd.println("\t<element name=\"top10vectorobjecten\"");
    xsd.println("\t type=\"tdn:top10vectorobjectenType\"");
    xsd.println("\t substitutionGroup=\"gml:_FeatureCollection\"/>\n");

    xsd.println("\t<element name=\"" + outputBaseName + "\" type=\"tdn:" +
featureTable + "Type\" substitutionGroup=\"gml:_Feature\"/>\n");

    xsd.println("\t<complexType name=\"top10vectorobjectenType\">");
    xsd.println("\t\t<complexContent>");
    xsd.println("\t\t\t<extension base=\"gml:AbstractFeatureCollectionType\">");
    xsd.println("\t\t\t\t<sequence>");
    xsd.println("\t\t\t\t\t<element ref=\"gml:complexProperty\" />");
    xsd.println("\t\t\t\t</sequence>");
    xsd.println("\t\t\t</extension>");
    xsd.println("\t\t</complexContent>");
    xsd.println("\t</complexType>\n");

    pS.println("<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>");
    pS.println("<!-- File: " + outputBaseName + ".xml  -->");

    pS.println("<tdn:top10vectorobjecten");
    pS.println(" xmlns:tdn=\"http://www.tdn.nl/top10test\"");
    pS.println(" xmlns:gml=\"http://www.opengis.net/gml\"");
    pS.println(" xmlns:xsi=\"http://www.w3.org/2000/10/XMLSchema-instance\"");
    pS.println(" xsi:schemaLocation=\"http://www.tdn.nl " + outputBaseName +
".xsd\">");
    pS.println("");
    pS.println("\t<gml:boundedBy>");
    pS.println("\t\t<gml:Box srsName=\"rd\">");
    pS.println("\t\t\t<gml:coordinates>0,0 300,300</gml:coordinates>");
    pS.println("\t\t</gml:Box>");
    pS.println("\t</gml:boundedBy>");
    pS.println("");


    gmlAdapter =
      OraSpatialManager.getGeometryAdapter("GML", "1.0",
                                           null, null, FileOutputStream.class);
    sdoAdapter =
      OraSpatialManager.getGeometryAdapter("SDO", "8.1.6",
                                           STRUCT.class, null, null, conn);


    getComplex((OracleConnection)conn);
    getFeature((OracleConnection)conn, "f");

    conn.close();
  } // End of method main()



  // Query object data table
  public static void getComplex(OracleConnection conn)
    throws Exception
  {
    //
    // Send query to the database.
    //
    pS.println("\t<gml:complexProperty>");

    String query = "SELECT * FROM " + complexBase + "_NODE";
    Statement stmt = conn.createStatement();
    OracleResultSet ors = (OracleResultSet)stmt.executeQuery(query);
    printPrimitive("p", ors);
    stmt.close();

    query = "SELECT * FROM " + complexBase + "_EDGE";
    stmt = conn.createStatement();
    ors = (OracleResultSet)stmt.executeQuery(query);
    printPrimitive("c", ors);
    stmt.close();

    query = "SELECT * FROM " + complexBase + "_RING";
    stmt = conn.createStatement();
    ors = (OracleResultSet)stmt.executeQuery(query);
    HashMap map = mapRings(ors);
    stmt.close();

    query = "SELECT * FROM " + complexBase + "_FACE";
    stmt = conn.createStatement();
    ors = (OracleResultSet)stmt.executeQuery(query);
    printPrimitive("s", ors, map);
    stmt.close();

    pS.println("\t</gml:complexProperty>");
  }


  //
  public static void printPrimitive(String ptype, OracleResultSet ors)
    throws Exception
  {
    printPrimitive(ptype, ors, null);
  }

  public static void printPrimitive(String ptype, OracleResultSet ors, HashMap map)
    throws Exception
  {
    String gid = null;

    ResultSetMetaData metadata =  ors.getMetaData();
    int ncolumns = metadata.getColumnCount();
    String columnNames[] = new String[ncolumns + 1];
    String columnNames2[] = new String[ncolumns + 1];

    int counter = 0;
    for (int j=1;j<=ncolumns;++j)
    {
   columnNames[j] = metadata.getColumnName(j);
        if ((columnNames[j]).equals("STARTNODE")) columnNames2[j] = new String
("startNode");
        else if ((columnNames[j]).equals("ENDNODE")) columnNames2[j] = new String
("endNode");
        else if ((columnNames[j]).equals("LEFTFACE")) columnNames2[j] = new String
("leftFace");
        else if ((columnNames[j]).equals("RIGHTFACE")) columnNames2[j] = new
String ("rightFace");
        else if ((columnNames[j]).equals("NCWEDGE")) columnNames2[j] = new String
("ncwEdge");
        else if ((columnNames[j]).equals("NAWEDGE")) columnNames2[j] = new String
("nawEdge");
        else if ((columnNames[j]).equals("STARTEDGE")) columnNames2[j] = new
String ("startEdge");
        else columnNames2[j] = columnNames[j].toLowerCase();
    }

    //
    // Write the XML file
    //
    System.out.println("Writing GML (" + "primitive " + ptype + ")");


    while (ors.next())
    {
      Geometry geom1 = null;
      StringBuffer attrB = new StringBuffer();
      StringBuffer rattrB = new StringBuffer();
      System.out.println("Geometry #" + (++counter));

      pS.println("\t\t<gml:primitive>");

      for (int i=1;i<=ncolumns;++i)
      {
    switch(metadata.getColumnType(i))
    {
      case Types.VARCHAR:
      case Types.NUMERIC:
        String output = ors.getString(i);
        if (output != null)
            {
              if (columnNames2[i].equals("id") || columnNames2[i].equals("gid"))
                { attrB.append(" " + columnNames2[i] + "=\"" + ptype + output + "\""); gid
= output; }
              else if (columnNames2[i].equals("startNode") ||
columnNames2[i].equals("endNode"))
                attrB.append(" " + columnNames2[i] + "=\"#p" + output + "\"");
              else if (columnNames2[i].equals("ncwEdge") ||
columnNames2[i].equals("nawEdge"))
                attrB.append(" " + columnNames2[i] + "=\"#c" + output + "\"");
              else if (columnNames2[i].equals("startEdge"))
                rattrB.append(" " + columnNames2[i] + "=\"#c" + output + "\"");
              else if (columnNames2[i].equals("leftFace") ||
columnNames2[i].equals("rightFace"))
                attrB.append(" " + columnNames2[i] + "=\"#s" + output + "\"");
              else
                attrB.append(" " + columnNames2[i] + "=\"" + output + "\"");
            }
        break;
      case 2002:
        STRUCT dbObject = (STRUCT) ors.getObject(i);
        geom1 = sdoAdapter.importGeometry(dbObject);
        break;
      default:
        System.out.println("Column " + i + " has type unkn: "
+metadata.getColumnTypeName(i) + "(" + metadata.getColumnType(i) + ")");
    }
      }
//HACK!!!!!      gmlAdapter.exportGeometry(pS, geom1);
      if (ptype.equals("p") || ptype.equals("c"))
      {
        pS.println(GMLGeometry.fromGeometryToGML(geom1, "\t\t\t", attrB.toString()));
      }
      else if (ptype.equals("s"))
      {
        pS.println("\t\t\t<gml:Polygon" + attrB.toString() + ">");
        pS.println("\t\t\t\t<outerBoundaryIs" + rattrB.toString() + "/>");

        ArrayList rings = (ArrayList) map.get(gid);
        if (rings != null)
        {
          int  nrings = rings.size();
          for (int j=0;j<nrings;++j)
          {
            pS.println("\t\t\t\t<innerBoundaryIs startEdge=\"#c" + rings.get(j) + "\"/>");
```

```
          }
        }

        pS.println("\t\t\t</gml:Polygon>");
      }
      pS.println("\t\t</gml:primitive>");
    }
  }


  // Query object data table
  public static void getFeature(OracleConnection conn, String flabel)
    throws Exception
  {
    //
    // Send query to the database.
    //
    String query = "SELECT * FROM " + featureTable;
    Statement stmt = conn.createStatement();
    OracleResultSet ors = (OracleResultSet)stmt.executeQuery(query);
    ResultSetMetaData metadata =  ors.getMetaData();

    //
    // Number of columns in table.
    //
    int ncolumns = metadata.getColumnCount();
    String columnNames[] = new String[ncolumns + 1];
    String columnNames2[] = new String[ncolumns + 1];

    for (int j=1;j<=ncolumns;++j)
    {
    columnNames[j] = metadata.getColumnName(j);
        if ((columnNames[j]).equals("POINTPROPERTY")) columnNames2[j] = new
String ("pointProperty");
        else if ((columnNames[j]).equals("CURVEPROPERTY")) columnNames2[j] =
new String ("curveProperty");
        else if ((columnNames[j]).equals("SURFACEPROPERTY")) columnNames2[j] =
new String ("surfaceProperty");
        else columnNames2[j] = columnNames[j].toLowerCase();
    }

    //
    // Write the XSD.
    //
    System.out.println("Writing XSD (" + featureTable + ")");



    xsd.println("\t<complexType name=\"" + featureTable + "Type\">");
    xsd.println("\t\t<complexContent>");
    xsd.println("\t\t\t<extension base=\"gml:AbstractFeatureType\">");
    xsd.println("\t\t\t\t<sequence>");

    for (int j=1;j<=ncolumns;++j)
    {
      switch(metadata.getColumnType(j))
      {
    case Types.VARCHAR:
        xsd.println(("\t\t\t\t\t<element name=\"" + columnNames2[j] + "\" type=\"string\"
/>"));
        break;
    case Types.NUMERIC:
            if (columnNames2[j].equals("fid") || columnNames2[j].equals("id"))
              ;
            else if (columnNames2[j].equals("pointProperty") ||
columnNames2[j].equals("curveProperty") ||
columnNames2[j].equals("surfaceProperty"))
          xsd.println(("\t\t\t\t\t<element name=\"" + columnNames2[j] + "\" type=\"" +
columnNames2[j] + "Type\" />"));
            else
          xsd.println(("\t\t\t\t\t<element name=\"" + columnNames2[j] + "\" type=\"integer\"
/>"));
        break;
    case 2002:
        xsd.println(("\t\t\t\t\t<element name=\"" + columnNames2[j] + "\"
type=\"gml:GeometryPropertyType\" />"));
        break;
    default:
        System.out.println("Column " + j + " has type unkn: "
+metadata.getColumnTypeName(j) + "(" + metadata.getColumnType(j) + ")");
      }
    }
    xsd.println(("\t\t\t\t</sequence>"));
    xsd.println(("\t\t\t</extension>"));
    xsd.println(("\t\t</complexContent>"));
    xsd.println(("\t</complexType>"));

    xsd.println(("</schema>"));

    //
    // Write the XML file
    //
    System.out.println("Writing GML (" + featureTable + ")");


    int counter = 0;

    while (ors.next())
    {
      StringBuffer fcontB = new StringBuffer();
      StringBuffer fattrB = new StringBuffer();

      System.out.println("Geometry #" + (++counter));

      for (int i=1;i<=ncolumns;++i)
      {
    switch(metadata.getColumnType(i))
    {
      case Types.VARCHAR:
      case Types.NUMERIC:
        String output = ors.getString(i);
            if (columnNames2[i].equals("fid") || columnNames2[i].equals("id"))
              fattrB.append(" " + columnNames2[i] + "=\"" + flabel + output + "\"");
            else if (columnNames2[i].equals("pointProperty"))
      fcontB.append("\t\t\t<gml:" + columnNames2[i] + " href=\"#p" + output + "\"/>\n");
            else if (columnNames2[i].equals("curveProperty"))
      fcontB.append("\t\t\t<gml:" + columnNames2[i] + " href=\"#c" + output + "\"/>\n");
            else if (columnNames2[i].equals("surfaceProperty"))
      fcontB.append("\t\t\t<gml:" + columnNames2[i] + " href=\"#s" + output + "\"/>\n");
        else if (output != null)
          fcontB.append("\t\t\t<tdn:" + columnNames2[i] + ">" + output + "</tdn:" +
columnNames2[i] + ">\n");
        else
          fcontB.append("\t\t\t<tdn:" + columnNames2[i] + " />");
        break;
      case 2002:
        fcontB.append("\t\t\t<tdn:" + columnNames2[i] + ">\n`");
        STRUCT dbObject = (STRUCT) ors.getObject(i);
        Geometry geom1 = sdoAdapter.importGeometry(dbObject);

//HACK!!!!!             gmlAdapter.exportGeometry(fcontB, geom1);
            fcontB.append(GMLGeometry.fromGeometryToGML(geom1, "\t\t\t\t",
fattrB.toString()));
        fcontB.append("\t\t\t</tdn:" + columnNames2[i] + ">");

        break;
      default:
        System.out.println("Column " + i + " has type unkn: "
+metadata.getColumnTypeName(i) + "(" + metadata.getColumnType(i) + ")");
    }
      }
      pS.println("\t<tdn:featureMember>");
      pS.println("\t\t<tdn:" + featureTable + fattrB.toString() + ">");
      pS.print(fcontB.toString());
      pS.println("\t\t</tdn:" + featureTable + ">");
      pS.println("\t</tdn:featureMember>");
    }
    pS.println("</tdn:top10vectorobjecten>");

  }

  public static HashMap mapRings(OracleResultSet ors)
    throws Exception
  {
    HashMap map = new HashMap();
    while (ors.next())
    {
      String sid = ors.getString(1);
      String sEdge = ors.getString(2);
      if (map.get(sid) == null)
        map.put(sid, new ArrayList());
      ((ArrayList) map.get(sid)).add(sEdge);
    }
    return map;
  }
}
```

D.2.2 SDOtoGML2.java

```
import java.io.*;
import java.sql.*;
import oracle.sql.STRUCT;
import oracle.jdbc.driver.*;
import oracle.sdoapi.OraSpatialManager;
import oracle.sdoapi.adapter.*;
import oracle.sdoapi.geom.*;



/**
 * This sample program reads all geometries from a database table and
 * converts them to the GML format.
 */
public class SDOtoGML2
{
  // static data members
  static String inputTable, outputBaseName;

  //
  // Convert SDO objects to GML
  //
  public static void main(String args[]) throws Exception
  {
    if (args.length != 1)
    {
      System.out.println("Parameters:");
      System.out.println("<Table name>: Table name of an input table");
      return;
    }

    outputBaseName = args[0];
    inputTable = args[0].toUpperCase();

    //
    // Connect to the database
    //
    Connection conn =

DriverManager.getConnection("jdbc:oracle:thin:@www.gdmc.nl:1521:igis","franklin",
"franklin");
    ((OracleConnection)conn).setDefaultRowPrefetch(100);

    // First have to register GML geometry adapter with the OraSpatialManager
    // because this adapter is not automatically registered
    // (in contrast to SDO, WKB, and WKT adapters)
    if (!OraSpatialManager.registerGeometryAdapter(
          new AdapterGML(OraSpatialManager.getGeometryFactory())))
    {
      System.out.println("The GML geometry adapter could not be registered.");
      return;
    }

    System.out.println("query");
    // Query for SDO_GEOMETRY object
    queryObject((OracleConnection)conn);

    conn.close();
  } // End of method main()

  // Query object data table
  public static void queryObject(OracleConnection conn)
    throws SQLException,
         InvalidGeometryException,
    IOException,
         GeometryInputTypeNotSupportedException,
    GeometryOutputTypeNotSupportedException
  {
    //
    // Send query to the database.
    //
    String query = "SELECT * FROM " + inputTable;
    Statement stmt = conn.createStatement();
    OracleResultSet ors = (OracleResultSet)stmt.executeQuery(query);
    ResultSetMetaData metadata =  ors.getMetaData();
```

```
    GeometryAdapter gmlAdapter =
      OraSpatialManager.getGeometryAdapter("GML", "1.0",
                                  null, null, FileOutputStream.class);
    GeometryAdapter sdoAdapter =
      OraSpatialManager.getGeometryAdapter("SDO", "8.1.6",
                                  STRUCT.class, null, null, conn);
    //
    // Number of columns in table.
    //
    int ncolumns = metadata.getColumnCount();
    String columnNames[] = new String[ncolumns + 1];

    for (int j=1;j<=ncolumns;++j)
    {
    columnNames[j] = metadata.getColumnName(j);
    }

    //
    // Write the XSD.
    //
    System.out.println("Writing XSD ...");
    PrintStream xsd = new PrintStream(new FileOutputStream(outputBaseName +
".xsd"));

    //
    // Write Stylesheet header.
    //
    xsd.println("<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\"?>");
    xsd.println("<!-- File: " + outputBaseName + ".xsd  -->");
    xsd.println("<schema");
    xsd.println(" targetNamespace=\"http://www.tdn.nl/top10test\"");
    xsd.println(" xmlns=\"http://www.w3.org/2000/10/XMLSchema\"");
    xsd.println(" xmlns:xlink=\"http://www.w3.org/1999/xlink\"");
    xsd.println(" xmlns:gml=\"http://www.opengis.net/gml\"");
    xsd.println(" xmlns:tdn=\"http://www.tdn.nl/top10test\"");
    xsd.println(" elementFormDefault=\"qualified\"");
    xsd.println(" version=\"0.4\">\n");

    xsd.println("\t<import namespace=\"http://www.opengis.net/gml\"
schemaLocation=\"feature.xsd\"/>");
    xsd.println("\t<element name=\"top10vectorobjecten\"");
    xsd.println("\t type=\"tdn:top10vectorobjectenType\"");
    xsd.println("\t substitutionGroup=\"gml:_FeatureCollection\"/>\n");

    xsd.println("\t<element name=\"" + outputBaseName + "\" type=\"tdn:" +
outputBaseName + "Type\" substitutionGroup=\"gml:_Feature\"/>\n");

    xsd.println("\t<complexType name=\"top10vectorobjectenType\">");
    xsd.println("\t\t<complexContent>");
    xsd.println("\t\t\t<extension base=\"gml:AbstractFeatureCollectionType\">");
    xsd.println("\t\t\t\t<sequence>");
    xsd.println("\t\t\t\t\t<element ref=\"gml:complexProperty\" />");
    xsd.println("\t\t\t\t</sequence>");
    xsd.println("\t\t\t</extension>");
    xsd.println("\t\t</complexContent>");
    xsd.println("\t</complexType>\n");

    xsd.println("\t<complexType name=\"" + outputBaseName + "Type\">");
    xsd.println("\t\t<complexContent>");
    xsd.println("\t\t\t<extension base=\"gml:AbstractFeatureType\">");
    xsd.println("\t\t\t\t<sequence>");


    for (int j=1;j<=ncolumns;++j)
    {
      switch(metadata.getColumnType(j))
      {
    case Types.VARCHAR:
      xsd.println(("\t\t\t\t\t<element name=\"" + columnNames[j] + "\" type=\"string\"
/>"));
      break;
    case Types.NUMERIC:
      xsd.println(("\t\t\t\t\t<element name=\"" + columnNames[j] + "\" type=\"integer\"
/>"));
      break;
    case 2002:
      xsd.println(("\t\t\t\t\t<element name=\"" + columnNames[j] + "\"
type=\"gml:GeometryPropertyType\" />"));
      break;
    default:
      System.out.println("Column " + j + " has type unkn: "
+metadata.getColumnTypeName(j) + "(" + metadata.getColumnType(j) + ")");
      }
    }
    xsd.println(("\t\t\t\t</sequence>"));
    xsd.println(("\t\t\t</extension>"));
    xsd.println(("\t\t</complexContent>"));
    xsd.println(("\t</complexType>"));

    xsd.println(("</schema>"));

    xsd.close();

    //
    // Write the XML file
    //
    System.out.println("Writing GML...");

    FileOutputStream oS = new FileOutputStream(outputBaseName + ".gml");
    PrintStream pS = new PrintStream(oS);

    pS.println("<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>");
    pS.println("<!-- File: " + outputBaseName + ".xml  -->");

    pS.println("<tdn:top10vectorobjecten");
    pS.println(" xmlns:tdn=\"http://www.tdn.nl/top10test\"");
    pS.println(" xmlns:gml=\"http://www.opengis.net/gml\"");
    pS.println(" xmlns:xsi=\"http://www.w3.org/2000/10/XMLSchema-instance\"");
    pS.println(" xsi:schemaLocation=\"http://www.tdn.nl " + outputBaseName +
".xsd\">");
    pS.println("");
    pS.println("\t<gml:boundedBy>");
    pS.println("\t\t<gml:Box srsName=\"rd\">");
    pS.println("\t\t\t<gml:coordinates>0,0 300,300</gml:coordinates>");
    pS.println("\t\t</gml:Box>");
    pS.println("\t</gml:boundedBy>");
    pS.println("");

    int counter = 0;
    while (ors.next())
    {
      System.out.println("Geometry #" + (++counter));

      pS.println("\t<tdn:featureMember>");
      pS.println("\t\t<tdn:" + inputTable + ">");
      for (int i=1;i<=ncolumns;++i)
      {
    switch(metadata.getColumnType(i))
    {
      case Types.VARCHAR:
      case Types.NUMERIC:
        String output = ors.getString(i);
        if (output != null)
        {
      pS.println("\t\t\t<tdn:"
            + columnNames[i]
        + ">"
        + output
        + "</tdn:"
        + columnNames[i]
        + ">");
        }
        else
        {
      pS.println("\t\t\t<tdn:" + columnNames[i] + " />");
        }
        break;
      case 2002:
        pS.println("\t\t\t<tdn:" + columnNames[i] + ">");
        STRUCT dbObject = (STRUCT) ors.getObject(i);
        Geometry geom1 = sdoAdapter.importGeometry(dbObject);

        gmlAdapter.exportGeometry(pS, geom1);
        pS.println("\n\t\t\t</tdn:" + columnNames[i] + ">");

        break;
      default:
        System.out.println("Column " + i + " has type unkn: "
+metadata.getColumnTypeName(i) + "(" + metadata.getColumnType(i) + ")");
    }
      }
      pS.println("\t\t</tdn:" + inputTable + ">");
      pS.println("\t</tdn:featureMember>");
    }
    pS.println("</tdn:top10vectorobjecten>");

    stmt.close();
    conn.close();
    oS.close();
    System.out.println("Done");
  }
}
```

D.2.3 GMLGeometry.java

```
import java.util.*;
import org.w3c.dom.*;
import oracle.sdoapi.adapter.*;
import oracle.sdoapi.geom.*;
import oracle.sdoapi.util.*;

/**
 * Defines the GML-based geometry.
 */
public class GMLGeometry
{
  private static int strIdx = 0;

  private static String srsName = "rd";

  public static String prefix = "gml";

  /**
   * From a gml geometry node generates a Geometry object.
   *
   * The GML format for geometries can be obtained from the following
   * URL: http://feature.opengis.org:8080/oracle.
   * example will be:
   * <pre>
   * <Polygon name="0" srsName="0">
   *   <LineString name="0" srsName="0">
   *   <CList>399682.8125,3861638.75 394632.125,3864919.25
390483.4375,3873216.5
   *         410129.6875,3874144.0 444692.0625,3873595.0 479631.9375,3872985.75
   *   </CList>
   *   </LineString>
   * </Polygon>
   * </pre>
   * @param nd  the DOM node
   * @return the oracle geometry with corresponding type
   */
  public static Geometry fromNodeToGeometry(Node nd, GeometryFactory gF)
    throws InvalidGeometryException, GeometryFormatException
  {
    String name = nd.getNodeName().toUpperCase();

    if (name.equals("POINT"))
      return fromNodeToPoint(nd, gF);
    else if (name.equals("LINESTRING"))
      return fromNodeToLineString(nd, gF);
    else if (name.equals("POLYGON"))
      return fromNodeToPolygon(nd, gF);
    else if (name.equals("MULTIPOINT"))
      return fromNodeToMultiPoint(nd, gF);
    else if (name.equals("MULTILINE"))
      return fromNodeToMultiLineString(nd, gF);
    else if (name.equals("MULTIPOLYGON"))
      return fromNodeToMultiPolygon(nd, gF);
    else if (name.equals("GEOMETRYCOLLECTION"))
      return fromNodeToGeometryCollection(nd, gF);
    else
      throw new InvalidGeometryException(ErrorMsg.get("GEOM-002") +
nd.getNodeName());
  }

  public static Point fromNodeToPoint(Node nd, GeometryFactory gF)
    throws InvalidGeometryException, GeometryFormatException
  {
    String coordStr;
    Node   clist;

    clist = nd.getFirstChild();
    if (clist == null)
      return null;
```

```
    int dimension = 2; //hard-coded here... LJ

    // the actual coordinates
    coordStr = clist.getFirstChild().getNodeValue();
    double[] cs = getCoordinates(dimension, coordStr);
    if(dimension==2)
      return gF.createPoint( cs[0], cs[1] );
    else
      return gF.createPoint( cs[0], cs[1], cs[2]);
  }

  public static LineString fromNodeToLineString(Node nd, GeometryFactory gF)
    throws InvalidGeometryException, GeometryFormatException
  {
    String coordStr;
    Node   clist;

    clist = nd.getFirstChild();
    if (clist == null)
      return null;
    int dimension = 2;  //hard-coded for now ... LJ

    // the actual coordinates
    coordStr = clist.getFirstChild().getNodeValue();
    return gF.createLineString(dimension, getCoordinates(dimension, coordStr));
  }

  public static Polygon fromNodeToPolygon(Node nd, GeometryFactory gF)
    throws InvalidGeometryException, GeometryFormatException
  {
    NodeList nl = nd.getChildNodes();
    LineString[] stringArray = new LineString[nl.getLength()];
    for (int i = 0; i < nl.getLength(); i++)
      stringArray[i] = fromNodeToLineString(nl.item(i), gF);

    return gF.createPolygon(stringArray);
  }

  public static MultiPoint fromNodeToMultiPoint(Node nd, GeometryFactory gF)
    throws InvalidGeometryException, GeometryFormatException
  {
    NodeList nl = nd.getChildNodes();
    Point[] pointArray = new Point[nl.getLength()];
    for (int i = 0; i < nl.getLength(); i++)
      pointArray[i] = fromNodeToPoint(nl.item(i), gF);

    return (MultiPoint)gF.createGeometryCollection(pointArray);
  }

  public static MultiLineString fromNodeToMultiLineString(Node nd, GeometryFactory
gF)
    throws InvalidGeometryException, GeometryFormatException
  {
    NodeList nl = nd.getChildNodes();
    LineString[] stringArray = new LineString[nl.getLength()];
    for (int i = 0; i < nl.getLength(); i++)
      stringArray[i] = fromNodeToLineString(nl.item(i), gF);

    return (MultiLineString)gF.createGeometryCollection(stringArray);
  }

  public static MultiPolygon fromNodeToMultiPolygon(Node nd, GeometryFactory gF)
    throws InvalidGeometryException, GeometryFormatException
  {
    NodeList nl = nd.getChildNodes();
    Polygon[] polyArray = new Polygon[nl.getLength()];
    for (int i = 0; i < nl.getLength(); i++)
      polyArray[i] = fromNodeToPolygon(nl.item(i), gF);

    return (MultiPolygon)gF.createGeometryCollection(polyArray);
  }

  public static GeometryCollection fromNodeToGeometryCollection(Node nd,
GeometryFactory gF)
    throws InvalidGeometryException, GeometryFormatException
  {
    NodeList nl = nd.getChildNodes();
    Geometry[] geomArray = new Geometry[nl.getLength()];
    for (int i = 0; i < nl.getLength(); i++)
      geomArray[i] = fromNodeToGeometry(nl.item(i), gF);

    return gF.createGeometryCollection(geomArray);
  }

  /**
   * get a double value start at 'idx' of the string.
   * !!! The string must have been trimmed. !!!
   */
  private static double getDoubleNumber(String in, int idx)
  {
    char   c;
    double  val = -9999;
    int     i = idx, j=idx, len = in.length();

    try
    {
      while( i < len && i >= 0)
      {
        c = in.charAt(i);
        if ( Character.isWhitespace(c) || c==',' )
          j=++i;
        else
          break;
      }

      while( i < len && i >= 0 )
      {
        c = in.charAt(i);
        if ( !Character.isWhitespace(c) && c!=',' )
          i++;
        else
          break;
      }

      if( (i>j && j>=0) || (i==j && i<len && i>=0) )
      {
        val = Double.valueOf(in.substring(j, i)).doubleValue();
        strIdx = i;
      }
    }
    catch (Exception e)
    {
      e.printStackTrace();
      strIdx = -1;
    }
    finally
    {
      if (i >= len)
        strIdx = -1;
      return val;
    }
  }

  private static double[] getCoordinates(int dimension, String in)
  {
    int     count=0, chunk=256;
    double  tmpArray[]=null, x, y, z=0;
    Vector  vec = new Vector(5);

    in = in.trim();
    strIdx = 0;
    while(strIdx >= 0) {
      x = getDoubleNumber(in, strIdx);
      y = getDoubleNumber(in, strIdx);
      if(dimension==3)
        z = getDoubleNumber(in, strIdx);

      if( (count%chunk)==0 ) {
        tmpArray = new double[chunk*dimension];
        vec.addElement(tmpArray);
      }

      tmpArray[(count%chunk)*dimension+0] = x;
      tmpArray[(count%chunk)*dimension+1] = y;
      if(dimension==3) tmpArray[(count%chunk)*dimension+2] = z;
      count ++;
    }
    //collapse all the chunks into a single array
    int i;
    tmpArray = new double[count*dimension];
    for(i=0; i<count/chunk; i++)
    {
      System.arraycopy((double[])vec.elementAt(i), 0, tmpArray,
                       i*chunk*dimension, chunk*dimension);
    }
    if( (count%chunk)!=0 ) //copy remaining coords
      System.arraycopy((double[])vec.elementAt(i), 0, tmpArray,
                       i*chunk*dimension, (count%chunk)*dimension);
    return tmpArray;
  }

  /**
   * Generates a string representing a DOM node for the given Geometry.
   */
  public static String fromGeometryToGML(Geometry geom)
  {
    return fromGeometryToGML(geom, "", "");
  }

  public static String fromGeometryToGML(Geometry geom, String indentation)
  {
    return fromGeometryToGML(geom, indentation, "");
  }

  public static String fromGeometryToGML(Geometry geom, String indentation, String
attr)
  {
    if (geom instanceof Point)
      return fromPointToGML((Point)geom, indentation, attr);
    else if (geom instanceof LineString)
      return fromLineStringToGML((LineString)geom, indentation, attr);
    else if (geom instanceof Polygon)
      return fromPolygonToGML((Polygon)geom, indentation, attr);
    else if (geom instanceof MultiPolygon)
      return fromMultiPolygonToGML((MultiPolygon)geom, indentation);
    else if (geom instanceof MultiLineString)
      return fromMultiLineStringToGML((MultiLineString)geom, indentation);
    else if (geom instanceof MultiPoint)
      return fromMultiPointToGML((MultiPoint)geom, indentation);
    else if (geom instanceof GeometryCollection)
      return fromGeometryCollectionToGML((GeometryCollection)geom, indentation);
    else return "";
  }

  /**
   *  Generate a GML coordinate pair.
   */
  public static String fromCoordToGML(double x,double y)
  {
    return new String("<gml:coord><gml:X>" + x + "</gml:X><gml:Y>" + y +
"</gml:Y></gml:coord>");
  }

  /**
   * Generates a string representing a DOM node for the given Point.
   */
  public static String fromPointToGML(Point p, String indentation, String attr)
  {
    return new String(indentation +
                      "<gml:Point" + attr + ">" +
          fromCoordToGML(p.getX(),p.getY()) +
          "</gml:Point>");
  }

  /**
   * Generates a string representing a DOM node out of the given line string.
   */

  public static String fromLineStringToGML(LineString ls, String indentation, String
attr)
  {
    StringBuffer strBuf = new StringBuffer(indentation + "<gml:LineString" + attr);
    double xArray[] = ls.getOrdArray(0);
    double yArray[] = ls.getOrdArray(1);
    if (xArray.length <= 2)
    {
      strBuf.append("/>");
    }
    else
    {
      strBuf.append(">\n");
      for(int i=0; i<xArray.length; i++)
      {
        strBuf.append(indentation + "\t" + fromCoordToGML(xArray[i],yArray[i]) + "\n");
      }

      strBuf.append(indentation + "</gml:LineString>");
    }
```

```
    return strBuf.toString();
  }

  /**
   * Generates a string representating a DOM node out of the given polygon.
   */

  public static String fromPolygonToGML(Polygon pg, String indentation, String attr)
  {
    StringBuffer strBuf = new StringBuffer(indentation + "<gml:Polygon srsName=\"" +
srsName + "\"" + attr + ">\n");
    strBuf.append(indentation + "\t<outerBoundaryIs>\n");
    strBuf.append(fromLinearRingToGML((LineString)pg.getExteriorRing(), indentation
+ "\t\t", ""));
    strBuf.append(indentation + "\t</outerBoundaryIs>\n");

    Enumeration ir = pg.getInteriorRings();
    while(ir.hasMoreElements())
    {
      strBuf.append(indentation + "\t<innerBoundaryIs>\n");
      strBuf.append(fromLinearRingToGML((LineString)ir.nextElement(), indentation +
"\t\t", ""));
      strBuf.append(indentation + "\t</innerBoundaryIs>\n");
    }
    strBuf.append(indentation + "</gml:Polygon>");
    return strBuf.toString();
  }

  /**
   * Generates a string representating a DOM node out of the linear ring
   */

  public static String fromLinearRingToGML(LineString ls, String indentation, String
attr)
  {
    StringBuffer strBuf = new StringBuffer(indentation + "<gml:LinearRing" + attr +
">\n");
    double xArray[] = ls.getOrdArray(0);
    double yArray[] = ls.getOrdArray(1);

    for(int i=0; i<xArray.length; i++)
    {
      strBuf.append(indentation + "\t" + fromCoordToGML(xArray[i],yArray[i]) + "\n");
    }
    strBuf.append(indentation + "</gml:LinearRing>\n");
    return strBuf.toString();
  }



  /**
   * Generates a string representating a DOM node out of the given multipolygon.
   */
  public static String fromMultiPolygonToGML(MultiPolygon mpg, String indentation)
  {
    StringBuffer strBuf = new StringBuffer(indentation + "<MultiPolygon>\n");
    Enumeration pg = mpg.getGeometries();
    while(pg.hasMoreElements())
    {
      strBuf.append(fromPolygonToGML((Polygon)pg.nextElement(), indentation + "\t",
""));
      strBuf.append("\n");
    }
    strBuf.append(indentation + "</MultiPolygon>\n");
    return strBuf.toString();
  }

  /**
   * Generates a string representating a DOM node out of the given multilinestring.
   */
  public static String fromMultiLineStringToGML(MultiLineString mls, String
indentation)
  {
    StringBuffer strBuf = new StringBuffer(indentation + "<MultiLine>\n");
    Enumeration ls = mls.getGeometries();
    while(ls.hasMoreElements())
    {
      strBuf.append(fromLineStringToGML((LineString)ls.nextElement(), indentation +
"\t", ""));
      strBuf.append("\n");
    }
    strBuf.append(indentation + "</MultiLine>\n");
    return strBuf.toString();
  }

  /**
   * Generates a string representating a DOM node out of the given multipoint.
   */
  public static String fromMultiPointToGML(MultiPoint mp, String indentation)
  {
    StringBuffer strBuf = new StringBuffer(indentation + "<MultiPoint>\n");
    Enumeration p = mp.getGeometries();
    while(p.hasMoreElements())
    {
      strBuf.append(fromPointToGML((Point)p.nextElement(), indentation + "\t", ""));
      strBuf.append("\n");
    }
    strBuf.append(indentation + "</MultiPoint>\n");
    return strBuf.toString();
  }

  /**
   * Generates a string representating a DOM node out of the given geometry
collection.
   */
  public static String fromGeometryCollectionToGML(GeometryCollection gc, String
indentation)
  {
    StringBuffer strBuf = new StringBuffer(indentation + "<GeometryCollection>\n");
    Enumeration g = gc.getGeometries();
    while(g.hasMoreElements())
    {
      strBuf.append(fromGeometryToGML((Geometry)g.nextElement(), indentation +
"\t"));
      strBuf.append("\n");
    }
    strBuf.append(indentation + "</GeometryCollection>\n");
    return strBuf.toString();
  }

  /**
   * Generates a string representing a DOM node out of an Envelope
   */

  public static String fromEnvelopeToGML(int dimension, Envelope env,
GeometryFactory gF)
    throws InvalidGeometryException
  {
    if (dimension != 2)
      throw new InvalidGeometryException(ErrorMsg.get("GEOM-001") + dimension);

    StringBuffer strBuf = new StringBuffer("<Envelope>");
    double ptsArray[] = new double[2*5];
    //CoordPoint   pts[]  = new CoordPoint[5];
    //for(int i=0; i<5; i++) pts[i] = new CoordPoint();

    ptsArray[0] = (env.getLowerLeft().getX());
    ptsArray[1] = (env.getLowerLeft().getY());
    ptsArray[2] = (env.getUpperRight().getX());
    ptsArray[3] = (env.getLowerLeft().getY());
    ptsArray[4] = (env.getUpperRight().getX());
    ptsArray[5] = (env.getUpperRight().getY());
    ptsArray[6] = (env.getLowerLeft().getX());
    ptsArray[7] = (env.getUpperRight().getY());
    ptsArray[8] = (env.getLowerLeft().getX());
    ptsArray[9] = (env.getLowerLeft().getY());

    //pts[0].setX(env.getLowerLeft().getX());
    //pts[0].setY(env.getLowerLeft().getY());
    //pts[1].setX(env.getUpperRight().getX());
    //pts[1].setY(env.getLowerLeft().getY());
    //pts[2].setX(env.getUpperRight().getX());
    //pts[2].setY(env.getUpperRight().getY());
    //pts[3].setX(env.getLowerLeft().getX());
    //pts[3].setY(env.getUpperRight().getY());
    //pts[4].setX(env.getLowerLeft().getX());
    //pts[4].setY(env.getLowerLeft().getY());

    LineString lr = gF.createLineString(dimension, ptsArray);
    //LinearRing   lr = new LinearRing(pts);
    Polygon pgn = gF.createPolygon(lr, null);
    strBuf.append(fromPolygonToGML(pgn, "", ""));
    strBuf.append("</Envelope>\n");
    return strBuf.toString();
  }

}
```

D.2.4 AdapterGML.java

```
// This is a straight copy from AdapterGML.java from the oracle
// sdo_api_examples.
//

import java.io.*;
import org.w3c.dom.*;
import oracle.sdoapi.OraSpatialManager;
import oracle.sdoapi.adapter.*;
import oracle.sdoapi.geom.*;
import oracle.sdoapi.sref.*;
import oracle.sdoapi.util.*;

public class AdapterGML implements GeometryAdapter
{
  protected final Class
    m_supportedInputTypes[] = { Node.class },
    m_supportedOutputTypes[] = { byte[].class, OutputStream.class },
    m_supportedPassthroughOutputTypes[] = { OutputStream.class };
  protected final String
    m_formatName = "GML",
    m_formatVersion = "1.0";

  protected GeometryFactory m_gF = null;

  public AdapterGML(GeometryFactory gFactory)
  {
    m_gF = gFactory;
  }

  public AdapterGML()
  {
    m_gF = OraSpatialManager.getGeometryFactory();
  }

  ////////////////////////////////////////////////////////////////////////
  // Converts Geometry objects
  public Object exportGeometry(Class outputType, Geometry geom)
    throws GeometryOutputTypeNotSupportedException, InvalidGeometryException
  {
    if (geom == null)
      return null;
    try
    {
      if (outputType.isAssignableFrom(byte[].class))
        return (GMLGeometry.fromGeometryToGML(geom)).getBytes();
      else if (outputType.isAssignableFrom(OutputStream.class))
      {
        OutputStream oStream = (OutputStream)outputType.newInstance();
        byte[] bytes = (GMLGeometry.fromGeometryToGML(geom)).getBytes();
        oStream.write(bytes);
        return oStream;
      }
      throw new GeometryOutputTypeNotSupportedException();
    }
    catch (Exception e)
    {
      throw new InvalidGeometryException();
    }
  }

  public void exportGeometry(Object outputObject, Geometry geom)
    throws GeometryOutputTypeNotSupportedException, InvalidGeometryException
  {
    if (geom == null)
      return;
    try
    {
      if (outputObject instanceof OutputStream)
      {
        byte[] bytes = (GMLGeometry.fromGeometryToGML(geom)).getBytes();
        ((OutputStream)outputObject).write(bytes);
      }
      else
        throw new GeometryOutputTypeNotSupportedException();
    }
    catch (Exception e)
```

```
  {
    throw new InvalidGeometryException();
  }
 }

 public Geometry importGeometry(Object inputSource)
  throws GeometryInputTypeNotSupportedException, InvalidGeometryException
 {
  if (inputSource == null)
    return null;
  if (!(inputSource instanceof Node))
    throw new GeometryInputTypeNotSupportedException();

  try
  {
    return GMLGeometry.fromNodeToGeometry((Node)inputSource, m_gF);
  }
  catch(Exception e)
  {
    throw new InvalidGeometryException();
  }
 }

 public Geometry importGeometry(Object inputSource, int nDim)
  throws GeometryInputTypeNotSupportedException, InvalidGeometryException
 {
  if (nDim == 2)
    return importGeometry(inputSource);
  else
    throw new InvalidGeometryException(ErrorMsg.get("GEOM-001") + nDim);
 }

 ////////////////////////////////////////////////////////////////////////////
 public boolean inputTypeSupported(Class inputFormat)
 { // is requested input format a subclass of what is expected?
  for(int i = 0; i < m_supportedInputTypes.length; i++)
    if((m_supportedInputTypes[i]).isAssignableFrom(inputFormat))
      return true;
  return false;
 }

 public boolean outputTypeSupported(Class outputFormat)
 { // is requested output format superclass of what can be provided?
  for(int i = 0; i < m_supportedOutputTypes.length; i++)
    if(outputFormat.isAssignableFrom((m_supportedOutputTypes[i])))
      return true;
  return false;
 }

 public boolean passthroughOutputTypeSupported(Class outputFormat)
 { // is requested output format superclass of what can be provided?
  for(int i = 0; i < m_supportedPassthroughOutputTypes.length; i++)
    if((m_supportedPassthroughOutputTypes[i]).isAssignableFrom(outputFormat))
      return true;
  return false;
 }

 public Class[] getSupportedInputTypes()
 {
  return m_supportedInputTypes;
 }

 public Class[] getSupportedOutputTypes()
 {
  return m_supportedOutputTypes;
 }

 public Class[] getSupportedPassthroughOutputTypes()
 {
  return m_supportedPassthroughOutputTypes;
 }

 public String getFormatName()
 {
  return m_formatName;
 }

 public String getFormatVersion()
 {
  return m_formatVersion;
 }

 public void setDefaultSRS(SpatialReference srs) { m_gF.setSpatialReference(srs); }

 public SpatialReference getDefaultSRS() { return m_gF.getSpatialReference(); }

}
```

D.3 QuickGis

D.3.1 LoaderGMLDom.java

```
package quak.gis;

import java.util.Vector;
import java.awt.*;
import java.awt.geom.*;
import quak.gis.*;


public class LoaderGMLDom extends Loader
{
   DomGMLReader gmlfile = null;

   public Vector load(
  Rectangle2D viewarea,
  int width,
  int height,
  String table,
  String attribute,
  String whereclause,
  Legenda legenda)
   {
  Vector objects = new Vector();

  try
  {
     if (gmlfile == null) gmlfile = new DomGMLReader(table,legenda);
     objects = gmlfile.getGeometries(attribute);
  }
  catch(Exception e)
  {
     System.err.println("Error reading file " + table + "\n(" + e + ")\n");
     e.printStackTrace();
     return(null);
  }
  return(objects);
   }

   public Rectangle2D getBbox(String tablename,String columnname)
   {
  try
  {
     if (gmlfile == null) gmlfile = new DomGMLReader(tablename);
     Rectangle2D rval = gmlfile.getBBox();
     return(rval);
  }
  catch(Exception e)
  {
     System.err.println("Error reading file " + tablename + "\n(" + e + ")\n");
     return(null);
  }
   }
}
```

D.3.2 LoaderGMLSax.java

```
package quak.gis;

import java.util.Vector;
import java.awt.*;
import java.awt.geom.*;
import quak.gis.*;


public class LoaderGMLSax extends Loader
{
   SaxGMLReader gmlfile = null;

   public Vector load(
  Rectangle2D viewarea,
  int width,
  int height,
  String table,
  String attribute,
  String whereclause,
  Legenda legenda)
   {
  Vector objects = new Vector();

  try
  {
     if (gmlfile == null) gmlfile = new SaxGMLReader(table,legenda);
     objects = gmlfile.getGeometries(attribute);
  }
  catch(Exception e)
  {
     System.err.println("Error reading file " + table + "\n(" + e + ")\n");
     e.printStackTrace();
     return(null);
  }
  return(objects);
   }

   public Rectangle2D getBbox(String tablename,String columnname)
   {
  try
  {
     if (gmlfile == null) gmlfile = new SaxGMLReader(tablename);
     Rectangle2D rval = gmlfile.getBBox();
     return(rval);
  }
  catch(Exception e)
  {
     System.err.println("Error reading file " + tablename + "\n(" + e + ")\n");
     return(null);
  }
   }
}
```

D.3.3 DomGMLReader.java

```
// Copyright (C) Oracle Corporation 1999.      All Rights Reserved.
package quak.gis;

import java.util.*;
import java.lang.Math;
import org.w3c.dom.*;
import oracle.sdoapi.adapter.*;
import oracle.sdoapi.geom.*;
import oracle.sdoapi.util.*;

import java.util.Vector;
import java.awt.*;
import java.awt.geom.*;


import org.apache.xerces.parsers.DOMParser;



/**
 * Defines the GML-based geometry.
 */
public class DomGMLReader
{
  // XML tree
  Node root;
  // geometry maps
  private HashMap pointmap = new HashMap();
  private HashMap curvemap = new HashMap();
  private HashMap surfacemap = new HashMap();
  private HashMap pointnmap = new HashMap();
  private HashMap curvenmap = new HashMap();
  private HashMap surfacenmap = new HashMap();
  // helper maps
```

```
private HashMap attrmap = new HashMap();
private HashMap coordmap = new HashMap();
private HashMap ccodenmap = new HashMap();
private HashMap descrnmap = new HashMap();
private HashMap refmap = new HashMap();
// rest
private String ccode = null;
private Legenda legenda = null;
private Color defcolor0 = new Color(0,0,0);
private Color defcolor1 = new Color(63,63,63);
private Color defcolor2 = new Color(127,127,127);
private double minx = Double.MAX_VALUE;
private double maxx = -(Double.MAX_VALUE/2);
private double miny = Double.MAX_VALUE;
private double maxy = -(Double.MAX_VALUE/2);
// for testing
private StringBuffer effe = null;


DomGMLReader(String xmlFile, Legenda legenda)
throws Exception
{
  this.legenda = legenda;
  ccode = legenda.getParamString();
  //  Create a Xerces DOM Parser
  DOMParser parser = new DOMParser();

  //  Parse the Document
  parser.parse(xmlFile);
  Document document = parser.getDocument();
  root = document.getDocumentElement();
  trim(root);
  findCodeRefDescr(root);
  findGeometries(root);
  clearHelperMaps();
}

DomGMLReader(String xmlFile)
throws Exception
{
  //  Create a Xerces DOM Parser
  DOMParser parser = new DOMParser();

  //  Parse the Document
  parser.parse(xmlFile);
  Document document = parser.getDocument();
  root = document.getDocumentElement();
  trim(root);
  findCodeRefDescr(root);
  findGeometries(root);
  clearHelperMaps();
}


public Vector getGeometries(String attr, Legenda l)
{
  legenda = l;
  return(getGeometries(attr));
}

public Vector getGeometries(String attr)
{
  if ((attr == null) || (attr.equalsIgnoreCase("")))
  {
    Vector objects = new Vector();
    objects.addAll(surfacemap.values());
    objects.addAll(curvemap.values());
    objects.addAll(pointmap.values());
    return objects;
  }
  else
  {
    return findAttributes(attr,root);
  }
}


public Rectangle2D getBBox()
{
  return new Rectangle2D.Double(minx,miny,maxx-minx,maxy-miny);
}

public Vector findAttributes(String attr, Node nd)
{
  Vector v = new Vector();
  findAttributesRec(v,attr,nd);
  return v;
}

private void findAttributesRec(Vector v, String attr, Node nd)
{
  String name = nd.getNodeName();
  if ((name.equalsIgnoreCase(attr)) || (name.equalsIgnoreCase("GML:" + attr)))
  {
    NamedNodeMap nnm = nd.getAttributes();
    Node refNode = nnm.getNamedItem("href");
    if (refNode != null)
    {
      String ref = refNode.getNodeValue().substring(1);
      if (surfacemap.get(ref) != null) v.add(surfacemap.get(ref));
      else if (curvemap.get(ref) != null) v.add(curvemap.get(ref));
      else if (pointmap.get(ref) != null) v.add(pointmap.get(ref));
    }
    else
    {
      if (surfacenmap.get(nd) != null) v.add(surfacenmap.get(nd));
      else if (curvenmap.get(nd) != null) v.add(curvenmap.get(nd));
      else if (pointnmap.get(nd) != null) v.add(pointnmap.get(nd));
    }
  }
  else if (nd.getNodeType() == Node.ELEMENT_NODE)
  {
    NodeList nl = nd.getChildNodes();
    for (int i=0; i<nl.getLength(); i++)
      findAttributesRec(v, attr, nl.item(i));
  }
}

private void findGeometries(Node nd)
{
  String name = nd.getNodeName();
  if (name.equalsIgnoreCase("#TEXT"))
    ;
  else if ((name.equalsIgnoreCase("Point")) ||
(name.equalsIgnoreCase("gml:Point")))
    fromNodeToPoint(nd);
  else if ((name.equalsIgnoreCase("LineString")) ||
(name.equalsIgnoreCase("gml:LineString")))
    fromNodeToLineString(nd);
  else if ((name.equalsIgnoreCase("Polygon")) ||
(name.equalsIgnoreCase("gml:Polygon")))
    fromNodeToPolygon(nd);
  else if (nd.getNodeType() == Node.ELEMENT_NODE)
  {
    NodeList nl = nd.getChildNodes();
    for (int i=0; i<nl.getLength(); i++)
      findGeometries(nl.item(i));
  }
}

private void findCodeRefDescr(Node nd)
{
  String name = nd.getNodeName().toUpperCase();
  NamedNodeMap nnm = nd.getAttributes();
  String code = null;
  Node refNode = null;
  String ref = null;

  // Code
  if (ccode != null) code = ccode.toUpperCase();
  if ((name.equals(code)) || (name.endsWith(":" + code)))
    ccodenmap.put(nd.getParentNode(),nd.getFirstChild().getNodeValue());

  // Ref
  if (nnm != null) refNode = nnm.getNamedItem("href");
  if (refNode != null)
  {
    ref = refNode.getNodeValue().substring(1);
    refmap.put(ref,nd.getParentNode());
  }

  // Descr
  if (nnm != null)
  {
    Node idNode = nnm.getNamedItem("id");
    if (idNode == null) idNode = nnm.getNamedItem("fid");
    if (idNode != null) descrnmap.put(nd,idNode.getNodeValue());
  }
  else if ((name.equals("NAME")) || (name.endsWith(":NAME")))
    descrnmap.put(nd.getParentNode(),nd.getFirstChild().getNodeValue());
  else if ((name.equals("DESCRIPTION")) || (name.endsWith(":DESCRIPTION")))
    descrnmap.put(nd.getParentNode(),nd.getFirstChild().getNodeValue());

  // Recursion
  if (nd.getNodeType() == Node.ELEMENT_NODE)
  {
    NodeList nl = nd.getChildNodes();
    for (int i=0; i<nl.getLength(); i++)
      findCodeRefDescr(nl.item(i));
  }
}

private void clearHelperMaps()
{
  attrmap = null;
  coordmap = null;
  ccodenmap = null;
  descrnmap = null;
  refmap = null;
}

private Point2D fromNodeToCoord(Node nd)
{
  NodeList cl = nd.getChildNodes();
  double x =
Double.valueOf(cl.item(0).getFirstChild().getNodeValue()).doubleValue();
  double y =
Double.valueOf(cl.item(1).getFirstChild().getNodeValue()).doubleValue();
  minx = Math.min(minx,x);
  miny = Math.min(miny,y);
  maxx = Math.max(maxx,x);
  maxy = Math.max(maxy,y);
  Point2D p2d = new Point2D.Double(x,y);

  return p2d;
}

private Point2D fromNodeToPoint(Node nd)
{
  NamedNodeMap nnm = nd.getAttributes();
  String id = nnm.getNamedItem("id").getNodeValue();
  if (id == null) id = nnm.getNamedItem("gid").getNodeValue();
  if (id != null) attrmap.put(id,nnm);

  Node coord = nd.getFirstChild();
  Point2D p2d = fromNodeToCoord(coord);
  String code = (String)ccodenmap.get(nd.getParentNode().getParentNode());
  if (code == null) code = (String)ccodenmap.get(refmap.get(id));
  Color color = defcolor0;
  if ((code != null) && (legenda != null)) color = legenda.getColor(code);
  String descr = (String)descrnmap.get(refmap.get(id));
  if (descr == null) descr =
(String)descrnmap.get(nd.getParentNode().getParentNode());
  MapPoint mp = new MapPoint(p2d,color,id + " " + descr);
  pointnmap.put(nd.getParentNode().getParentNode(),mp);
  if (id != null)
  {
    coordmap.put(id,p2d);
    pointmap.put(id,mp);
//System.err.println ("DomGMLReader:" + id + " " + color);
  }
  return p2d;
}

private GeneralPath fromNodeToLineString(Node nd)
{
  GeneralPath p = new GeneralPath(GeneralPath.WIND_NON_ZERO);
  fromNodeToLineString(nd, p);
  return p;
}

private GeneralPath fromNodeToLineString(Node nd, GeneralPath p)
{
  ArrayList l = new ArrayList();
  Point2D p2d = null;
  NamedNodeMap nnm = nd.getAttributes();
  String id = nnm.getNamedItem("id").getNodeValue();
```

```
    if (id == null) id = nnm.getNamedItem("gid").getNodeValue();
    if (id != null) attrmap.put(id,nnm);
    String startNode = nnm.getNamedItem("startNode").getNodeValue().substring(1);
    String endNode = nnm.getNamedItem("endNode").getNodeValue().substring(1);
    Node leftNode = nnm.getNamedItem("leftFace");
    Node rightNode = nnm.getNamedItem("rightFace");
    String leftFace = "-";
    String rightFace = "-";
    if (leftNode != null) leftFace =  leftNode.getNodeValue().substring(1);
    if (rightNode != null) rightFace = rightNode.getNodeValue().substring(1);

    NodeList nl = nd.getChildNodes();
    if (nl.getLength() == 0)
    {
      if ((startNode!=null) && (endNode!=null))
      {
        p2d = (Point2D)coordmap.get(startNode);
        l.add(p2d);
        p.moveTo((float)p2d.getX(),(float)p2d.getY());
        p2d = (Point2D)coordmap.get(endNode);
        l.add(p2d);
        p.lineTo((float)p2d.getX(),(float)p2d.getY());
      }
    }
    else
    {
      p2d = fromNodeToCoord(nl.item(0));
      l.add(p2d);
      p.moveTo((float)p2d.getX(),(float)p2d.getY());
      for (int i = 1; i < nl.getLength(); i++)
      {
        p2d = fromNodeToCoord(nl.item(i));
        l.add(p2d);
        p.lineTo((float)p2d.getX(),(float)p2d.getY());
      }
    }
    String code = (String)ccodenmap.get(nd.getParentNode().getParentNode());
    if (code == null) code = (String)ccodenmap.get(refmap.get(id));
    Color color = defcolor1;
    if ((code != null) && (legenda != null)) color = legenda.getColor(code);
    String descr = (String)descrnmap.get(refmap.get(id));
    if (descr == null) descr =
(String)descrnmap.get(nd.getParentNode().getParentNode());
    MapLineString ml = new MapLineString(p,color,id + " " + descr);
    curvenmap.put(nd.getParentNode().getParentNode(),ml);
    if (id != null);
    {
      coordmap.put(id,l);
      curvemap.put(id,ml);
//System.out.println ("DomGMLReader:" + id + " " + startNode + endNode + " " +
leftFace + rightFace + " " + l.size());
    }
    return p;
  }

  private GeneralPath fromNodeToLinearRing(Node nd)
  {
    GeneralPath p = fromNodeToLineString(nd);
    p.closePath();
    return p;
  }

  private GeneralPath fromNodeToLinearRing(Node nd, GeneralPath p)
  {
    fromNodeToLineString(nd,p);
    p.closePath();
    return p;
  }

  private GeneralPath fromNodeToPolygon(Node nd)
  {
effe = new StringBuffer();
    GeneralPath p = new GeneralPath(GeneralPath.WIND_NON_ZERO);
    NamedNodeMap nnm = nd.getAttributes();
    String id = nnm.getNamedItem("id").getNodeValue();
    if (id == null) id = nnm.getNamedItem("gid").getNodeValue();
    if (id != null) attrmap.put(id,nnm);

    NodeList nl = nd.getChildNodes();
    NamedNodeMap bnnm = nl.item(0).getAttributes();
    String startEdge = bnnm.getNamedItem("startEdge").getNodeValue().substring(1);
    if (startEdge != null)
    {
      reconstructLinearRing(p,id,startEdge,0);
      for (int i = 1; i < nl.getLength(); i++)
      {
        bnnm = nl.item(i).getAttributes();
        startEdge = bnnm.getNamedItem("startEdge").getNodeValue().substring(1);
        reconstructLinearRing(p,id,startEdge,1);
      }
    }
    else
    {
      fromNodeToLinearRing(nl.item(0).getFirstChild(),p);
      for (int i = 1; i < nl.getLength(); i++)
      {
        fromNodeToLinearRing(nl.item(i).getFirstChild(),p);
      }
    }
    String code = (String)ccodenmap.get(nd.getParentNode().getParentNode());
    if (code == null) code = (String)ccodenmap.get(refmap.get(id));
    Color color = defcolor2;
    if ((code != null) && (legenda != null)) color = legenda.getColor(code);
    String descr = (String)descrnmap.get(refmap.get(id));
    if (descr == null) descr =
(String)descrnmap.get(nd.getParentNode().getParentNode());
    MapPolygon mp = new MapPolygon(p,color,id + " " + descr);
    surfacenmap.put(nd.getParentNode().getParentNode(),mp);
    if (id != null)
    {
      surfacemap.put(id,mp);
//System.out.println ("DomGMLReader:" + id + effe.toString());
    }
    return p;
  }

  private void reconstructLinearRing(GeneralPath p, String id, String startEdge, int
orientation)
  {
effe.append(" ");
    String next = null;
    String rightFace = null;
    String leftFace = null;
    NamedNodeMap edgeAttr = (NamedNodeMap)attrmap.get(startEdge);
    Node rightNode = edgeAttr.getNamedItem("rightFace");
    Node leftNode = edgeAttr.getNamedItem("leftFace");
    if (rightNode!=null) rightFace = rightNode.getNodeValue().substring(1);
    if (leftNode!=null) leftFace = leftNode.getNodeValue().substring(1);
    if (id.equalsIgnoreCase(rightFace))
    {
      setpath(p,startEdge,id,orientation);
      next = edgeAttr.getNamedItem("ncwEdge").getNodeValue().substring(1);
    }
    else if (id.equalsIgnoreCase(leftFace))
    {
      setpath(p,startEdge,id,orientation);
      next = edgeAttr.getNamedItem("nawEdge").getNodeValue().substring(1);
    }
    while (!(startEdge.equalsIgnoreCase(next)))
    {
      rightFace = null;
      leftFace = null;
      edgeAttr = (NamedNodeMap)attrmap.get(next);
      rightNode = edgeAttr.getNamedItem("rightFace");
      leftNode = edgeAttr.getNamedItem("leftFace");
      if (rightNode!=null) rightFace = rightNode.getNodeValue().substring(1);
      if (leftNode!=null) leftFace = leftNode.getNodeValue().substring(1);
      if (id.equalsIgnoreCase(rightFace))
      {
        append(p,next,id,orientation);
        next =
((NamedNodeMap)attrmap.get(next)).getNamedItem("ncwEdge").getNodeValue().su
bstring(1);
      }
      else if (id.equalsIgnoreCase(leftFace))
      {
        append(p,next,id,orientation);
        next =
((NamedNodeMap)attrmap.get(next)).getNamedItem("nawEdge").getNodeValue().su
bstring(1);
      }
    }
  }


  private void setpath(GeneralPath p, String edge, String face, int orientation)
  {
effe.append(edge);
    String rightFace = null;
    String leftFace = null;
    NamedNodeMap edgeAttr = (NamedNodeMap)attrmap.get(edge);
    Node rightNode = edgeAttr.getNamedItem("rightFace");
    Node leftNode = edgeAttr.getNamedItem("leftFace");
    if (rightNode!=null) rightFace = rightNode.getNodeValue().substring(1);
    if (leftNode!=null) leftFace = leftNode.getNodeValue().substring(1);

    ArrayList l = (ArrayList)coordmap.get(edge);
    if (face.equalsIgnoreCase(rightFace))
    {
      Point2D p2d = (Point2D)l.get(0);
      p.moveTo((float)p2d.getX(),(float)p2d.getY());
      for(int i=1; i<l.size(); i++)
      {
        p2d = (Point2D)l.get(i);
        p.lineTo((float)p2d.getX(),(float)p2d.getY());
      }
    }
    else if (face.equalsIgnoreCase(leftFace))
    {
      Point2D p2d = (Point2D)l.get(l.size()-1);
      p.moveTo((float)p2d.getX(),(float)p2d.getY());
      for(int i=1; i<l.size(); i++)
      {
        p2d = (Point2D)l.get(l.size()-i-1);
        p.lineTo((float)p2d.getX(),(float)p2d.getY());
      }
    }
  }

  private void append(GeneralPath p, String edge, String face, int orientation)
  {
effe.append(edge);
    String rightFace = null;
    String leftFace = null;
    NamedNodeMap edgeAttr = (NamedNodeMap)attrmap.get(edge);
    Node rightNode = edgeAttr.getNamedItem("rightFace");
    Node leftNode = edgeAttr.getNamedItem("leftFace");
    if (rightNode!=null) rightFace = rightNode.getNodeValue().substring(1);
    if (leftNode!=null) leftFace = leftNode.getNodeValue().substring(1);

    ArrayList l = (ArrayList)coordmap.get(edge);
    if (face.equalsIgnoreCase(rightFace))
    {
      for(int i=0; i<l.size(); i++)
      {
        Point2D p2d = (Point2D)l.get(i);
        p.lineTo((float)p2d.getX(),(float)p2d.getY());
      }
    }
    else if (face.equalsIgnoreCase(leftFace))
    {
      for(int i=0; i<l.size(); i++)
      {
        Point2D p2d = (Point2D)l.get(l.size()-i-1);
        p.lineTo((float)p2d.getX(),(float)p2d.getY());
      }
    }
  }

  private void trim(Node root) {
    boolean ascending = false;
    Node previousChecker = null;

    Node checker=root;
    while (true) {
      //*** TAKE ACTION ON NODE WITH CHECKER ***
      if ((!ascending) && (checker.getNodeType() == Node.TEXT_NODE)) {
          String trimmedText = checker.getNodeValue().trim();
          if (trimmedText.equalsIgnoreCase("")) {
            checker.getParentNode().removeChild(checker);
            checker=previousChecker; //back to undeleted node
          }
        }
      previousChecker=checker;

      //*** GO DOWN IF YOU CAN ***
      if ((checker.hasChildNodes()) && (!ascending)) {
        checker=checker.getFirstChild();
```

```
        ascending = false;
      }
      //*** OTHERWISE GO RIGHT IF YOU CAN ***
      else if (checker.getNextSibling() != null) {
        checker=checker.getNextSibling();
        ascending = false;
      }
      //*** OTHERWISE GO UP IF YOU CAN ***
      else if (checker.getParentNode() != null) {
        checker=checker.getParentNode();
        ascending = true;
      }
      //*** OTHERWISE YOU'VE ASCENDED BACK TO ***
      //*** THE DOCUMENT ELEMENT, SO YOU'RE DONE ***
      else {
        break;
      }
    }
  }


}
```

D.3.4 SaxGMLReader.java

```
// Copyright (C) Oracle Corporation 1999.       All Rights Reserved.
package quak.gis;

import java.util.*;
import java.io.*;
import java.lang.Math;
import org.w3c.dom.*;
import oracle.sdoapi.adapter.*;
import oracle.sdoapi.geom.*;
import oracle.sdoapi.util.*;

import java.util.Vector;
import java.awt.*;
import java.awt.geom.*;

//import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import org.apache.xerces.parsers.*;

import franklin.gis.*;

/**
 * Defines the GML-based geometry.
 */
public class SaxGMLReader
{
  private MyCollection mycollection = null;
  private Rectangle2D bbox = null;
  private String ccode = null;
  private Legenda legenda = null;
  private Color defcolor0 = new Color(0,0,0);
  private Color defcolor1 = new Color(63,63,63);
  private Color defcolor2 = new Color(127,127,127);


  SaxGMLReader(String uri, Legenda legenda)
  throws Exception
  {
    this.legenda = legenda;
    ccode = legenda.getParamString();
    read(uri);
  }

  SaxGMLReader(String uri)
  throws Exception
  {
    read(uri);
  }


  private void read(String uri)
  throws Exception
  {

    XMLReader xmlReader = null;
    try {
        xmlReader = new SAXParser();
    } catch (Exception ex) {
        System.err.println(ex);
    }

    // Set the ContentHandler of the XMLReader
    SaxGMLHandler handler = new SaxGMLHandler();
    xmlReader.setContentHandler(handler);

    xmlReader.parse(uri);

    mycollection = handler.getCollection();
    bbox = handler.getBBox();
  }


  public Vector getGeometries(String attr, Legenda l)
  {
    legenda = l;
    ccode = legenda.getParamString();
    return(getGeometries(attr));
  }

  public Vector getGeometries(String attr)
  {
    Vector objects = new Vector();
    java.util.List primitives = mycollection.getList();
    if (primitives==null) return objects;
    String attr2 = "";
    String feat2 = "";
    int index = attr.indexOf(".");
    if (index>0)
    {
      feat2 = attr.substring(0,index).trim();
      attr2 = attr.substring(index+1).trim();
    }
    else
    {
      attr2 = attr.trim();
    }
    for (int i=0;i<primitives.size();i++)
    {
      MyPrimitive g = (MyPrimitive)primitives.get(i);
      MyFeature f = g.getFeature();
      if ((attr2.equals("")) || (attr2.equalsIgnoreCase(g.getRole())))
      {
        if ((feat2.equals("")) || (f!=null) && (feat2.equalsIgnoreCase(f.getType())))
        {
          String code = null;
          if (f!=null) code = f.getProperty(ccode);
          String id = g.getId();
          String descr = g.toString();
          Color color = null;
          if (code!=null)
            color = legenda.getColor(code);
          if (g instanceof MyPoint)
          {
            if (color==null) color = defcolor0;
            MapPoint mp = new MapPoint(((MyPoint)g).getPoint(),color,descr);
            objects.add(mp);
          }
          else if (g instanceof MyCurve)
          {
            if (color==null) color = defcolor1;
            MapLineString ml = new
MapLineString(((MyCurve)g).getCurve(),color,descr);
            objects.add(ml);
          }
          else if (g instanceof MySurface)
          {
            if (color==null) color = defcolor2;
            MapPolygon mp = new
MapPolygon(((MySurface)g).getSurface(),color,descr);
            objects.add(mp);
          }
        }
      }
    }
    return objects;
  }


  public Rectangle2D getBBox()
  {
    return bbox;
  }


}
```

D.3.5 franklin.gis package

```
package franklin.gis;


public class MyCollection
{
  private java.util.List primitives = new java.util.ArrayList();


  public void add(MyPrimitive p)
  {
    primitives.add(p);
  }

  public java.util.List getList()
  {
    return(primitives);
  }
}
package franklin.gis;


public class MyComplex extends MyPrimitive
{
  private java.util.List nodes = new java.util.ArrayList();
  private java.util.List edges = new java.util.ArrayList();
  private java.util.List faces = new java.util.ArrayList();

  public void addNode(MyNode n)
  {
    nodes.add(n);
  }

  public void addEdge(MyEdge e)
  {
    edges.add(e);
  }

  public void addFace(MyFace f)
  {
    faces.add(f);
  }

  public java.util.List getNodeList()
  {
    return(nodes);
  }

  public java.util.List getEdgeList()
  {
    return(edges);
  }

  public java.util.List getFaceList()
  {
    return(faces);
  }
}
package franklin.gis;

import java.awt.geom.*;

public class MyCurve extends MyEdge
{
  private java.util.List coords = null;


  public MyCurve( String gid)
```

```
  {
    setId(gid);
  }

  public java.util.List getPList()
  {
    if (coords==null)
    {
      java.util.List plist = new java.util.ArrayList();
      MyPoint start = (MyPoint)getStartNode();
      MyPoint end = (MyPoint)getEndNode();
      if ((start!=null) && (end!=null))
      {
        plist.add(start.getPoint().clone());
        plist.add(end.getPoint().clone());
      }
      return plist;
    }
    else
      return coords;
  }

  public void addPoint(Point2D p)
  {
    if (coords==null) coords = new java.util.ArrayList();
    coords.add(p.clone());
  }

  public void addPoint(double x, double y)
  {
    if (coords==null) coords = new java.util.ArrayList();
    coords.add(new Point2D.Double(x,y));
  }

  public GeneralPath getPath()
  {
    GeneralPath gp = new GeneralPath();
    java.util.List plist = getPList();
    if (plist.size()>=2)
    {
      Point2D p = (Point2D)plist.get(0);
      gp.moveTo((float)p.getX(),(float)p.getY());
      for (int i=1;i<plist.size();i++)
      {
        p = (Point2D)plist.get(i);
        gp.lineTo((float)p.getX(),(float)p.getY());
      }
    }
    return gp;
  }

  public GeneralPath getCurve()
  {
    return getPath();
  }

}
package franklin.gis;


public class MyEdge extends MyPrimitive
{
  private MyNode startNode;
  private MyNode endNode;
  private MyFace leftFace;
  private MyFace rightFace;
  private MyEdge ncwEdge;
  private MyEdge nawEdge;
  private MyEdge pcwEdge;
  private MyEdge pawEdge;


  public MyEdge()
  {
  }

  public MyEdge(String s)
  {
    setId(s);
  }

  public void setStartNode(MyNode n) { startNode = n; }
  public void setEndNode(MyNode n) { endNode = n; }
  public void setLeftFace(MyFace f) { leftFace = f; }
  public void setRightFace(MyFace f) { rightFace = f; }
  public void setNcwEdge(MyEdge e) { ncwEdge = e; }
  public void setNawEdge(MyEdge e) { nawEdge = e; }
  public MyNode getStartNode() { return startNode; }
  public MyNode getEndNode() { return endNode; }
  public MyFace getLeftFace() { return leftFace; }
  public MyFace getRightFace() { return rightFace; }
  public MyEdge getNcwEdge() { return ncwEdge; }
  public MyEdge getNawEdge() { return nawEdge; }

  public String toString()
  {
    StringBuffer buf = new StringBuffer() ;
    if (getId()!=null) buf.append("edge " + getId());
    else buf.append("edge");
    buf.append (" (");
    if (startNode!=null) buf.append("sN=" + startNode.getId());
    if (endNode!=null) buf.append(" eN=" + endNode.getId());
    if (leftFace!=null) buf.append(" lF=" + leftFace.getId());
    if (rightFace!=null) buf.append(" rF=" + rightFace.getId());
    if (ncwEdge!=null) buf.append(" ncE=" + ncwEdge.getId());
    if (nawEdge!=null) buf.append(" naE=" + nawEdge.getId());
    if (pcwEdge!=null) buf.append(" pcE=" + pcwEdge.getId());
    if (pawEdge!=null) buf.append(" paE=" + pawEdge.getId());
    buf.append (")");
    if (getFeature()!=null) buf.append ("\t" + getFeature());
    if (getRole()!=null) buf.append("\trole " + getRole());
//    buf.append ("\n");
    return buf.toString();
  }
}
package franklin.gis;

import java.awt.geom.*;

public class MyFace extends MyPrimitive
{
  private MyEdge outerRing;
  private java.util.List innerRings = new java.util.ArrayList();

  public MyFace()
  {
  }

  public MyFace(String s)
  {
    setId(s);
  }

  public void setStartEdge(MyEdge e)
  {
    outerRing = e;
  }
  public void addStartEdge(MyEdge e)
  {
    innerRings.add(e);
  }

  public MyEdge getStartEdge()
  {
    return(outerRing);
  }
  public java.util.List getStartEdgeList()
  {
    return(innerRings);
  }

  public String toString()
  {
    StringBuffer buf = new StringBuffer() ;
    if (getId()!=null) buf.append("face " + getId());
    else buf.append("face");
    buf.append (" (");

    if (outerRing!=null) buf.append("osE=" + outerRing.getId());
    if (innerRings.size()>=1)
    {
      buf.append(" iE=");
      if ((MyEdge)innerRings.get(0)!=null)
buf.append(((MyEdge)innerRings.get(0)).getId());
    }
    for (int i=1;i<innerRings.size();i++)
    {
      if ((MyEdge)innerRings.get(i)!=null) buf.append( "," +
((MyEdge)innerRings.get(i)).getId());
    }
    buf.append (")");
    if (getFeature()!=null) buf.append ("\t" + getFeature());
    if (getRole()!=null) buf.append("\trole " + getRole());
//    buf.append ("\n");
    return buf.toString();
  }

}
package franklin.gis;


public class MyFeature
{
  private String fid;
  private String typ;
  private java.util.Map props = new java.util.HashMap();

  public MyFeature(String type, String id)
  {
    if (type!=null) typ = new String(type);
    if (id!=null) fid = new String(id);
  }

  public String getType()
  {
    return typ;
  }

  public java.util.Map getProperties()
  {
    return props;
  }

  public String getProperty(String name)
  {
    String s = ((String)props.get(name));
    if (s==null) s = ((String)props.get(name.toLowerCase()));
    return s;
  }

  public void setProperty(String name, Object value)
  {
    props.put(name.toLowerCase(),value);
  }

  public String toString()
  {
    StringBuffer buf = new StringBuffer();
    if (fid!=null) buf.append(typ + " " + fid);
    else buf.append(typ);

    java.util.Iterator it = props.keySet().iterator();
    buf.append (" (");
    if (it.hasNext())
    {
      String key = (String)it.next();
      String value = (String)props.get(key);
      buf.append(key + "=" + value);
    }
    while (it.hasNext())
    {
      String key = (String)it.next();
      String value = (String)props.get(key);
      buf.append(" " + key + "=" + value);
    }
    buf.append (")");
//    buf.append("\n");
    return buf.toString();
  }
}
package franklin.gis;


public class MyNode extends MyPrimitive
{
  private MyFace containingFace;
```

```
  public MyNode()
  {
  }

  public MyNode(String id)
  {
    setId(id);
  }

  public MyFace getContainingFace() { return containingFace; }
  public void setContainingFace(MyFace f) { containingFace = f; }

  public String toString()
  {
    StringBuffer buf = new StringBuffer() ;
    if (getId()!=null) buf.append("node " + getId());
    else buf.append("node");
    if (containingFace!=null)
      buf.append(" (cF=" + containingFace.getId() + ")");
    if (getFeature()!=null) buf.append("\t" + getFeature());
    if (getRole()!=null) buf.append("\trole " + getRole());
//    buf.append ("\n");
    return buf.toString();
  }

}
package franklin.gis;

import java.awt.geom.*;

public class MyPoint extends MyNode
{
  private Point2D p;

  public MyPoint(String gid)
  {
    setId(gid);
  }

  public void setPoint(Point2D pnt)
  {
    p = (Point2D)pnt.clone();
  }

  public void setPoint(double x, double y)
  {
    p = new Point2D.Double(x,y);
  }

  public Point2D getPoint()
  {
    return p;
  }
}
package franklin.gis;


public class MyPrimitive extends Object
{
  private String id;
  private MyFeature feature;
  private String role;

  public MyPrimitive()
  {
    id = null;
  }

  public MyPrimitive(String id)
  {
    if (id!=null)
      this.id = new String(id);
    else
      this.id = null;
  }

  public String getId() { return id; }
  public MyFeature getFeature() { return feature; }
  public String getRole() { return role; }
  public void setId(String id) { if (id == null) this.id = null; else this.id = new String(id); }
  public void setFeature(MyFeature f) { feature = f; }
  public void setRole(String role) { this.role = new String(role); }

  public String toString()
  {
    StringBuffer buf = new StringBuffer();
    if (id!=null) buf.append("geometry " + id);
    else buf.append("geometry");
    if (feature!=null) buf.append("\t" + feature);
    if (role!=null) buf.append("\trole " + role);
//    buf.append("\n");
    return buf.toString();
  }
}
package franklin.gis;

import java.awt.geom.*;

public class MySurface extends MyFace
{
  private GeneralPath s = null;

  public MySurface(String gid)
  {
    setId(gid);
  }

  public GeneralPath getPath()
  {
    if (s==null) return deriveSurface();
    else return s;
  }

  public GeneralPath getSurface()
  {
    return getPath();
  }

  public void addRing(MyCurve c)
  {
    if (s==null) s = new GeneralPath();
    addRing(s,c);
  }

  public void addRing(java.util.List l)
  {
    if (s==null) s = new GeneralPath();
    addRing(s,l);
  }

  private void addRing(GeneralPath gp, MyCurve c)
  {
    java.util.List l = c.getPList();
    addRing(gp, l);
  }

  private void addRing(GeneralPath gp, java.util.List l)
  {
    Point2D p = (Point2D)l.get(0);
    gp.moveTo((float)p.getX(),(float)p.getY());
    for (int i=1;i<l.size();i++)
    {
      p = (Point2D)l.get(i);
      gp.lineTo((float)p.getX(),(float)p.getY());
    }
    gp.closePath();
  }

  private GeneralPath deriveSurface()
  {
    GeneralPath gp = new GeneralPath();
    MyCurve start = (MyCurve)getStartEdge();
    if (start!=null)
    {
      addRing(gp, deriveRing(start));
      java.util.List l = getStartEdgeList();
      for (int i=1;i<l.size();i++)
      {
        start = (MyCurve)l.get(i);
        addRing(gp, deriveRing(start));
      }
    }
    return(gp);
  }

  private java.util.List deriveRing(MyCurve c)
  {
    MyCurve next = null;
    java.util.List ring = new java.util.ArrayList();
    java.util.List plist = c.getPList();
    if (this == c.getRightFace())
    {
      for (int i=0;i<plist.size();i++)
      {
        ring.add((Point2D)plist.get(i));
      }
      next = (MyCurve)c.getNcwEdge();
    }
    else if (this == c.getLeftFace())
    {
      for (int i=0;i<plist.size();i++)
      {
        ring.add((Point2D)plist.get(plist.size()-i-1));
      }
      next = (MyCurve)c.getNawEdge();
    }
    while (c != next)
    {
      plist = next.getPList();
      if (this == next.getRightFace())
      {
        for (int i=0;i<plist.size();i++)
        {
          ring.add((Point2D)plist.get(i));
        }
        next = (MyCurve)next.getNcwEdge();
      }
      else if (this == next.getLeftFace())
      {
        for (int i=0;i<plist.size();i++)
        {
          ring.add((Point2D)plist.get(plist.size()-i-1));
        }
        next = (MyCurve)next.getNawEdge();
      }
    }
    return ring;
  }

}
package franklin.gis;

import java.io.*;
import java.util.*;
import java.lang.Math;
import java.awt.geom.*;
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;

public class SaxGMLHandler extends DefaultHandler{

  List elems;
  List feats;

  MyComplex mycomplex;
  MyCollection mycollection;
  MyPrimitive myprimitive;
  MyFeature myfeature;
  MyPoint mypoint;
  MyCurve mycurve;
  MySurface mysurface;
  double x;
  double y;
  double z;
  String value;

  Map objMap;
  Map startMap;
  Map endMap;
  Map leftMap;
  Map rightMap;
  Map ncwMap;
  Map nawMap;
  Map pcwMap;
  Map pawMap;
  Map outerMap;
```

```
  Map innerMap;
  Map fidMap;
  Map refMap;
  Map roleMap;

  List featList;

  String gid;
  String startNode;
  String endNode;
  String leftFace;
  String rightFace;
  String ncwEdge;
  String nawEdge;
  String startEdge;

  String fid;
  String ref;

  String elem;
  String parentElem;
  String previousElem;

  double minx;
  double maxx;
  double miny;
  double maxy;

  int strIdx;


  public MyCollection getCollection()
  {
    return(mycollection);
  }

  public Rectangle2D getBBox()
  {
    return(new Rectangle2D.Double(minx,miny,maxx-minx,maxy-miny));
  }

  public void startDocument()
  throws SAXException
  {
System.err.println("SaxGMLHandler:startDocument");
    mycomplex = new MyComplex();
    mycollection = new MyCollection();
    myprimitive = null;
    myfeature = null;
    mypoint = null;
    mycurve = null;
    mysurface = null;
    value = null;

    elems = new ArrayList();
    feats = new ArrayList();
    objMap = new HashMap();
    startMap = new HashMap();
    endMap = new HashMap();
    leftMap = new HashMap();
    rightMap = new HashMap();
    ncwMap = new HashMap();
    nawMap = new HashMap();
    pcwMap = new HashMap();
    pawMap = new HashMap();
    outerMap = new HashMap();
    innerMap = new HashMap();
    fidMap = new HashMap();
    refMap = new HashMap();
    roleMap = new HashMap();

    featList = new ArrayList();

    minx = Double.MAX_VALUE;
    maxx = -(Double.MAX_VALUE/2);
    miny = Double.MAX_VALUE;
    maxy = -(Double.MAX_VALUE/2);

    gid = null;
    startNode = null;
    endNode = null;
    leftFace = null;
    rightFace = null;
    ncwEdge = null;
    nawEdge = null;
    startEdge = null;
    fid = null;
    ref = null;
    elem = null;
    parentElem = null;
    previousElem = null;
  }

  public void endDocument()
  throws SAXException
  {
    linkGeometry();
    linkReference();
    List l = mycollection.getList();
    l.addAll(mycomplex.getFaceList());
    l.addAll(mycomplex.getEdgeList());
    l.addAll(mycomplex.getNodeList());
    removeEmptyFeatures();
System.err.println("SaxGMLHandler:endDocument:" +
mycomplex.getNodeList().size() + " points + " + mycomplex.getEdgeList().size() + "
curves + " + mycomplex.getFaceList().size() + " surfaces = " +
mycollection.getList().size() + " total");
  }

  public void startElement(String uri, String localName, String qName, Attributes atts)
  throws SAXException
  {
    parentElem = elem;
    elem = localName;
    elems.add(0,elem);
    value = null;

    if( localName.equalsIgnoreCase("Point"))
    {
      gid = null;
      gid = atts.getValue("gid");
      if (gid == null) gid = atts.getValue("id");
      mypoint = new MyPoint(gid);
      mypoint.setFeature((MyFeature)feats.get(1));
      mypoint.setRole(parentElem);
      mycomplex.addNode(mypoint);
      if (gid!=null)
        objMap.put(gid,mypoint);
    }
    else if( localName.equalsIgnoreCase("LineString"))
    {
      gid = null;
      gid = atts.getValue("gid");
      if (gid == null) gid = atts.getValue("id");
      startNode = atts.getValue("startNode");
      endNode = atts.getValue("endNode");
      leftFace = atts.getValue("leftFace");
      rightFace = atts.getValue("rightFace");
      ncwEdge = atts.getValue("ncwEdge");
      nawEdge = atts.getValue("nawEdge");
      mycurve = new MyCurve(gid);
      mycurve.setFeature((MyFeature)feats.get(1));
      mycurve.setRole(parentElem);
      mycomplex.addEdge(mycurve);
      if (gid!=null)
      {
        objMap.put(gid,mycurve);
        if (startNode!=null) startMap.put(gid,startNode.substring(1));
        if (endNode!=null) endMap.put(gid,endNode.substring(1));
        if (leftFace!=null) leftMap.put(gid,leftFace.substring(1));
        if (rightFace!=null) rightMap.put(gid,rightFace.substring(1));
        if (ncwEdge!=null) ncwMap.put(gid,ncwEdge.substring(1));
        if (nawEdge!=null) nawMap.put(gid,nawEdge.substring(1));
      }
    }
    else if( localName.equalsIgnoreCase("LinearRing"))
    {
      gid = null;
      gid = atts.getValue("gid");
      if (gid == null) gid = atts.getValue("id");
      startNode = atts.getValue("startNode");
      endNode = atts.getValue("endNode");
      leftFace = atts.getValue("leftFace");
      rightFace = atts.getValue("rightFace");
      ncwEdge = atts.getValue("ncwEdge");
      nawEdge = atts.getValue("nawEdge");
      mycurve = new MyCurve(gid);
      if (! ((parentElem.equalsIgnoreCase("outerBoundaryIs"))
        || (parentElem.equalsIgnoreCase("innerBoundaryIs"))))
      {
        mycurve.setFeature((MyFeature)feats.get(1));
        mycurve.setRole(parentElem);
        mycomplex.addEdge(mycurve);
      }
      if (gid!=null)
      {
        objMap.put(gid,mycurve);
        if (startNode!=null) startMap.put(gid,startNode.substring(1));
        if (endNode!=null) endMap.put(gid,endNode.substring(1));
        if (leftFace!=null) leftMap.put(gid,leftFace.substring(1));
        if (rightFace!=null) rightMap.put(gid,rightFace.substring(1));
        if (ncwEdge!=null) ncwMap.put(gid,ncwEdge.substring(1));
        if (nawEdge!=null) nawMap.put(gid,nawEdge.substring(1));
      }
    }
    else if( localName.equalsIgnoreCase("Polygon"))
    {
      gid = null;
      gid = atts.getValue("gid");
      if (gid == null) gid = atts.getValue("id");
      mysurface = new MySurface(gid);
      mysurface.setFeature((MyFeature)feats.get(1));
      mysurface.setRole(parentElem);
      mycomplex.addFace(mysurface);
      if (gid!=null)
        objMap.put(gid,mysurface);
    }

    else if( localName.equalsIgnoreCase("outerBoundaryIs"))
    {
      mycurve = null;
      startEdge = atts.getValue("startEdge");
      if (startEdge!=null) outerMap.put(mysurface,startEdge.substring(1));
    }
    else if( localName.equalsIgnoreCase("innerBoundaryIs"))
    {
      mycurve = null;
      startEdge = atts.getValue("startEdge");
      if (startEdge!=null) {
        List list = (List)innerMap.get(mysurface);
        if (list==null) list = new ArrayList();
        list.add(startEdge.substring(1));
        innerMap.put(mysurface,list);
      }
    }

    else if( localName.equalsIgnoreCase("coord"))
    {
    }

    else if( localName.equalsIgnoreCase("coordinates"))
    {
    }


    else
    {
      gid = null;
      fid = atts.getValue("fid");
      if (fid==null) fid = atts.getValue("id");
      ref = atts.getValue("href");
      if (ref!=null)
      {
        refMap.put(ref.substring(1),myfeature);
        roleMap.put(ref.substring(1),elem);
      }
      myfeature = new MyFeature(elem,fid);
      feats.add(0,myfeature);
      featList.add(myfeature);
      if (fid!=null) fidMap.put(fid,myfeature);
    }
  }

  public void endElement(String uri, String localName, String qName)
  throws SAXException
  {

    if( elem.equalsIgnoreCase("Point"))
```

```
    {
    }
    else if( elem.equalsIgnoreCase("LineString"))
    {
    }
    else if( elem.equalsIgnoreCase("LinearRing"))
    {
    }
    else if( elem.equalsIgnoreCase("Polygon"))
    {
    }

    else if( elem.equalsIgnoreCase("innerBoundaryIs"))
    {
      if (mycurve!=null) mysurface.addRing(mycurve);
    }
    else if( elem.equalsIgnoreCase("outerBoundaryIs"))
    {
      if (mycurve!=null) mysurface.addRing(mycurve);
    }

    else if( elem.equalsIgnoreCase("coord"))
    {
      if (parentElem.equalsIgnoreCase("Point"))
        mypoint.setPoint(x,y);
      else if(parentElem.equalsIgnoreCase("LineString"))
        mycurve.addPoint(x,y);
      else if(parentElem.equalsIgnoreCase("LinearRing"))
        mycurve.addPoint(x,y);
    }

    else if( elem.equalsIgnoreCase("coordinates"))
    {
      if (parentElem.equalsIgnoreCase("Point"))
        getCoordinates(mypoint,2,value);
      else if(parentElem.equalsIgnoreCase("LineString"))
        getCoordinates(mycurve,2,value);
      else if(parentElem.equalsIgnoreCase("LinearRing"))
        getCoordinates(mycurve,2,value);
    }

    else
    {
      feats.remove(0);
      if (feats.size() > 1)
      {
        myfeature = (MyFeature)feats.get(0);
        if (value!=null)
        {
          myfeature.setProperty(elem,value);
        }
      }
      value = null;
    }


    previousElem = elem;
    elems.remove(0);
    if (elems.size() == 0)
      elem = null;
    else
      elem = (String)elems.get(0);
    if (elems.size()<=1)
      parentElem = null;
    else
      parentElem = (String)elems.get(1);
  }

  public void characters(char[] ch, int start, int length)
  throws SAXException
  {
    String s = new String(ch,start,length);
    String v = null;
    if (s!=null) v= s.trim();
    if( elem.equalsIgnoreCase("X"))
    {
      x = Double.valueOf(v).doubleValue();
      minx = Math.min(minx,x);
      maxx = Math.max(maxx,x);
    }
    else if( elem.equalsIgnoreCase("Y"))
    {
      y = Double.valueOf(v).doubleValue();
      miny = Math.min(miny,y);
      maxy = Math.max(maxy,y);
    }
    else if( elem.equalsIgnoreCase("Z"))
    {
      z = Double.valueOf(v).doubleValue();
    }
    else if((v!=null) && (!v.equals("")))
    {
      value = v;
    }
    else
    {
      value = null;
    }
  }


  private void getCoordinates(MyPrimitive prim, int dimension, String in)
  {
    double  x=0, y=0, z=0;

    in = in.trim();
    strIdx = 0;
    while(strIdx >= 0) {
      x = getDoubleNumber(in, strIdx);
      y = getDoubleNumber(in, strIdx);
      if(dimension==3)
        z = getDoubleNumber(in, strIdx);
      minx = Math.min(minx,x);
      maxx = Math.max(maxx,x);
      miny = Math.min(miny,y);
      maxy = Math.max(maxy,y);
      if (prim instanceof MyPoint)
        ((MyPoint)prim).setPoint(x,y);
      else if (prim instanceof MyCurve)
        ((MyCurve)prim).addPoint(x,y);
    }
  }

  private double getDoubleNumber(String in, int idx)
  {
    char    c;
    double  val = -9999;
    int     i = idx, j=idx, len = in.length();

    try
    {
      while( i < len && i >= 0)
      {
        c = in.charAt(i);
        if ( Character.isWhitespace(c) || c==',' )
          j=++i;
        else
          break;
      }

      while( i < len && i >= 0 )
      {
        c = in.charAt(i);
        if ( !Character.isWhitespace(c) && c!=',' )
          i++;
        else
          break;
      }

      if( (i>j && j>=0) || (i==j && i<len && i>=0) )
      {
        val = Double.valueOf(in.substring(j, i)).doubleValue();
        strIdx = i;
      }
    }
    catch (Exception e)
    {
      e.printStackTrace();
      strIdx = -1;
    }
    if (i >= len)
      strIdx = -1;
    return val;
  }

  private void linkGeometry()
  {
System.err.println("SaxGMLHandler:linkGeometry");
    Iterator it = objMap.keySet().iterator();
    while (it.hasNext())
    {
      String key = (String)it.next();
      Object o = null;
      MyPrimitive prim = (MyPrimitive)objMap.get(key);
      if (prim instanceof MyNode)
      {
        MyNode n = (MyNode)prim;
      }
      else if (prim instanceof MyEdge)
      {
        MyEdge e = (MyEdge)prim;
        e.setStartNode((MyNode)objMap.get((String)startMap.get(key)));
        e.setEndNode((MyNode)objMap.get((String)endMap.get(key)));
        e.setLeftFace((MyFace)objMap.get((String)leftMap.get(key)));
        e.setRightFace((MyFace)objMap.get((String)rightMap.get(key)));
        e.setNcwEdge((MyEdge)objMap.get((String)ncwMap.get(key)));
        e.setNawEdge((MyEdge)objMap.get((String)nawMap.get(key)));
      }
      else if (prim instanceof MyFace)
      {
        MyFace f = (MyFace)prim;
        MyEdge e = (MyEdge)objMap.get((String)outerMap.get(f));
        f.setStartEdge(e);
        List l = (List)innerMap.get(f);
        if (l!=null)
        {
          for (int i=0;i<l.size();i++)
          {
            e = (MyEdge)objMap.get((String)l.get(i));
            f.addStartEdge(e);
          }
        }
      }
    }
  }

  private void linkReference()
  {
System.err.println("SaxGMLHandler:linkReference");
    Iterator it = refMap.keySet().iterator();
    while (it.hasNext())
    {
      String key = (String)it.next();
      MyPrimitive prim = (MyPrimitive)objMap.get(key);
      MyFeature feat = (MyFeature)refMap.get(key);
      if (prim!=null)
      {
        prim.setFeature(feat);
        prim.setRole((String)roleMap.get(key));
      }
    }
  }

  private void removeEmptyFeatures()
  {
System.err.println("SaxGMLHandler:removeEmptyFeature");
    List l = mycollection.getList();
    for (int i=0; i<l.size();i++)
    {
      MyPrimitive prim = (MyPrimitive)l.get(i);
      MyFeature feat = prim.getFeature();
      Map props = null;
      if (feat!=null) props = feat.getProperties();
      if (props==null) prim.setFeature(null);
      else if (props.size()==0) prim.setFeature(null);
    }
  }
}
```