
Architecture Specification

Project number:	IST-2000-30090
Project title:	GiMoDig
Deliverable number:	D4.4.1
Deliverable type:	PU
Deliverable nature:	Specification
Contributing WP:	WP 4
Contractual date of delivery:	February 28, 2003
Actual date of delivery:	May 22, 2003
Authors:	Finnish Geodetic Institute (FGI) / Lassi Lehto
Keywords:	Service Architecture, WMS, WFS, GML, OpenLS

The logo for GiMoDig is written in a blue, stylized, cursive font. The letter 'D' is significantly larger and contains a small, colorful map of Finland.

Abstract: This document describes the GiMoDig System Architecture. As stated in the project proposal, the GiMoDig project aims at establishing a standards-based infrastructure through which a mobile user can access up-to-date geospatial data resources from National Mapping Agencies' (NMA) databases.

The GiMoDig architecture is based on a layered service stack, in which a service would make queries to the service below it, do some processing on the data received as a response, and provide the results of its processing as a service to the service layer above it. Five-level system architecture is proposed. In the first level the NMAs would run a Data Service providing raw spatial data in an XML –encoded form. Above the data services is the Data Integration layer. The responsibilities of this layer include coordinate transformations to a common reference frame and schema transformations to the GiMoDig Global Schema. The third layer is called the Data Processing layer, and is responsible for the generalisation of spatial data. The fourth layer in the system architecture is called Portal Service. The main responsibilities of this layer consist of providing a basic metadata service to the client, processing the service requests coming from the client, forwarding the request in an appropriate form to the GiMoDig Application Service layer, and transforming the resulting map according to the capabilities of the client platform in question. On the fifth layer are the client applications.

The communication from layer to layer is to be based on internationally accepted interface standards. The main interface specifications to be implemented in the GiMoDig architecture are: 1. Web Feature Service (WFS) specification of the OGC as the access interface in the Data Service layer, 2. Web Map Service (WMS) specification of the OGC as the query interface of the Portal Layer, 3. Presentation Service of the Open Location Services initiative as an alternative access interface on the Portal Layer. A GiMoDig–specific service interface, that so far does not have a widely accepted specification, is the access interface to the Generalisation Service.

Contents

EXECUTIVE SUMMARY	3
1 INTRODUCTION	4
1.1 Overview	4
1.2 Layered Architecture	7
1.3 Service Dialog.....	9
1.4 Standardised Interfaces	10
1.5 Data Encoding	13
2 DATA SERVICE LAYER.....	16
2.1 Overview	16
2.2 WFS Implementation.....	17
3 DATA INTEGRATION LAYER.....	20
3.1 Overview	20
3.2 Detailed Description.....	21
4 DATA PROCESSING LAYER.....	22
4.1 Overview	22
4.2 Internal Structure	22
5 PORTAL SERVICE LAYER.....	23
5.1 Overview	23
5.2 Detailed Structure	24
6 CLIENT LAYER.....	25
BIBLIOGRAPHY AND REFERENCES	27
APPENDIX A	29
APPENDIX B	34
APPENDIX C	38

Executive Summary

Overview

This report proposes a five-tier system architecture for the project 'Geospatial info-mobility service by real-time data-integration and generalisation' (GiMoDig). The layers of the architecture are: Data Service layer, Data Integration layer, Data Processing layer, Portal Service layer and Client layer. Each of the layers have a clearly defined role in the overall service functionality. Communication between the layers is based on standardised interfaces. The main interface specifications to be implemented in the GiMoDig architecture include the Web Feature Service (WFS) specification of the Open GIS Consortium (OGC) as the access interface on the Data Service layer and the Data Integration layer, Web Map Service (WMS) specification of the OGC as the query interface on the Portal Layer, and the Presentation Service of the Open Location Services initiative as an alternative access interface on the Portal Layer. One open interface, Generalisation Service, is to be designed as part of the project.

Service Layers

The Data Services on the first layer provide raw spatial data in an XML-encoded form from the NMAs' databases. The Data Integration service on the next layer of the architecture plays an important role in the GiMoDig project. The main task of the Data Integration layer is to provide a single access point to the topographic data sources of the participating NMAs. From the Data Integration layer the data is available in a harmonised data model, the GiMoDig Global Schema, and in a common, ETRS89-based coordinate reference system. The responsibilities of this layer thus include tasks like coordinate transformation and schema transformation.

Above the Data Integration layer is the Data Processing layer. The main function of this layer is data generalisation. In the generalisation process the harmonisation needs of the datasets and the mobile client device characteristics are to be considered. The fourth layer in the architecture is called Portal Service. The main tasks of this layer include providing a basic metadata service to the client, and transforming the map created by the service to match the capabilities of the client platform.

On the fifth layer are the client applications. The goal is to make the GiMoDig service available for a set of different client environments. These include traditional Web browsing on a PC platform, SVG viewer-based clients on PDA devices, and Java clients on mobile phones.

Acknowledgements

Several persons have contributed to this architecture specification. Comments and suggestions provided by Tapani Sarjakoski, the coordinator of the GiMoDig project, have had a significant influence on the development of the specification. The architecture has been discussed in various GiMoDig project meetings, in which representatives from all GiMoDig consortium members have given valuable input. Architectural issues have also been dealt with in the two technical meetings of the WP 7, Real-time Generalisation.

D4.4.1

Architecture Specification

1 Introduction

This document constitutes the final plan of the System Architecture for the project 'Geospatial info-mobility service by real-time data-integration and generalisation' (GiMoDig), (Sarjakoski et al, 2002; GiMoDig, 2003). The architecture of the GiMoDig service is to be based on a multi-tiered processing model in a networked environment. In this architecture primary geodata is retrieved directly from the data provider's database at the moment the service is accessed. In most of the current existing service solutions massive a-priori data transfers are needed from data provider to service provider, creating unnecessary data duplication and making data updates difficult. In the GiMoDig system architecture spatial data is maintained in the NMA's server and provided to the network through a standardised access interface. The communication between actors on various levels of the system hierarchy is to be based on the emerging spatial Web-standards, especially those developed by the Open GIS Consortium (OGC, 2003). The project aims at applying the Web-related spatial standards, being developed by the OGC and other standards setting bodies, in the broadest possible sense.

The encoding of spatial data being transferred across the network from database level to the middle-layer service is going to be based on the Extensible Markup Language (XML,2000) technology. The OGC has defined an XML application for spatial data, called Geography Markup Language (GML, 2003). Many GIS software vendors are currently developing support for this new data encoding mechanism into their products.

The aim of this document is to define the general architecture for the eventual testbed prototype service to be developed in Work Packages 6 (Common access interface) and 8 (Integration and prototyping). The main questions to be answered in this WP include the balancing of the processing responsibilities between the individual data services, the middleware service and the client application. There are many possible architectures varying from the fat client approach, in which the client accesses the data services directly and does all the integration and visualisation processing locally, to the thin client architecture where all the processing is done in the middleware service and only the resulting map image is sent to the client. An appropriately designed architecture should facilitate the system to serve a wide array of mobile end user devices with varying characteristics, and at the same time also provide support for traditional Web browsing.

1.1 Overview

The GiMoDig architecture is based on a layered service stack, in which a service would make queries to the service below it, do some processing on the data received as a response, and provide the results of this processing as a service to the layer above it. The level of detail in specifying the layers is a matter of discussion, but if the services were to

be run on separate computers communicating through a network, too fine-grained service definition would create a significant disadvantage in terms of overall system performance.

For the above-mentioned reason, a five-level system architecture is proposed (Sarjakoski and Lehto, 2003). In the first level each NMA runs a Data Service providing raw spatial data in an XML-encoded form. These services are to be based on a common interface specification to facilitate the task of the layer above, the Data Integration layer.

The Data Integration layer plays a prominent role in the GiMoDig context, as data integration is one of the main focus areas of the project. The main task of the Data Integration layer is to provide a single access point to the topographic data sources of the participating NMAs – serving that data in a common data model (the GiMoDig Global Schema) and in a common coordinate reference system (ETRS89). The responsibilities of this layer thus include tasks like coordinate transformations and schema transformations. Some generalisation procedures will also be needed in the harmonisation of the datasets.

Above the Data Integration layer is the Data Processing layer. In the case of the GiMoDig project the main interest on this layer is focused on data generalisation. The main factors to be considered in the generalisation process include the mobile client device characteristics, user preferences and the application for which the map is going to be used.

The fourth layer in the system architecture is called Portal Service. The main responsibilities of this layer consist of providing a basic metadata service to the client, processing the service requests coming from the client, forwarding the request in an appropriate form to the GiMoDig Data Processing service layer, and transforming the resulting map according to the capabilities of the client platform in question.

On the fifth layer are the client applications. The idea in the GiMoDig service is to make the resulting service available for a set of different client environments. At the very minimum the following three client platforms are to be supported: 1. Traditional Web browsing on a PC platform, 2. SVG viewer-based clients on PDA devices, 3. Java Mobile Information Device Profile (Java MIDP, 2003) clients on mobile phones. As indicated in the report of the WP 3 (Small-display cartography), only the more advanced 'Smart phones' can be considered as GiMoDig client devices (Nissen et al, 2003).

The communication from layer to layer is to be based on internationally accepted interface standards. At the moment the main interface specifications to be implemented in the GiMoDig architecture are: 1. Web Feature Service (WFS, 2002) specification of the OGC as the access interface in the Data Service layer and the Data Integration layer, 2. Web Map Service (WMS, 2002) specification of the OGC as the query interface of the Portal Layer, 3. Presentation Service of the Open Location Services (OpenLS) initiative as an alternative access interface on the Portal Layer (OpenLS, 2003). A GiMoDig-specific service interface, that so far does not have a widely accepted specification is the Generalisation Service on the Data Processing layer.

As the standardised interfaces are to be applied to different layers of the service architecture, the system provides a flexible setup in which clients do not necessarily need to go through all the levels of the hierarchy, but can rather communicate with the service that provides best support for the task in hand. More powerful clients might thus consult the WFS interfaces directly to process GML data, whereas some client applications might prefer to communicate with Generalisation Service, and the most restricted platforms might require the client application to rely on the standard WMS service providing raster images.

The data encoding between the Data Service and the Application Service layers is to be based on the Geography Markup Language (GML) as the WFS specification dictates. The version of GML to be applied in the GiMoDig service is the current official release 3.0. The visual representation of the spatial data is mainly to be based on the different profiles of the Scalable Vector Graphics (SVG, 2001) vector format.

A schematic model of the system architecture, as presented in the GiMoDig project contract, is shown in Figure 1.

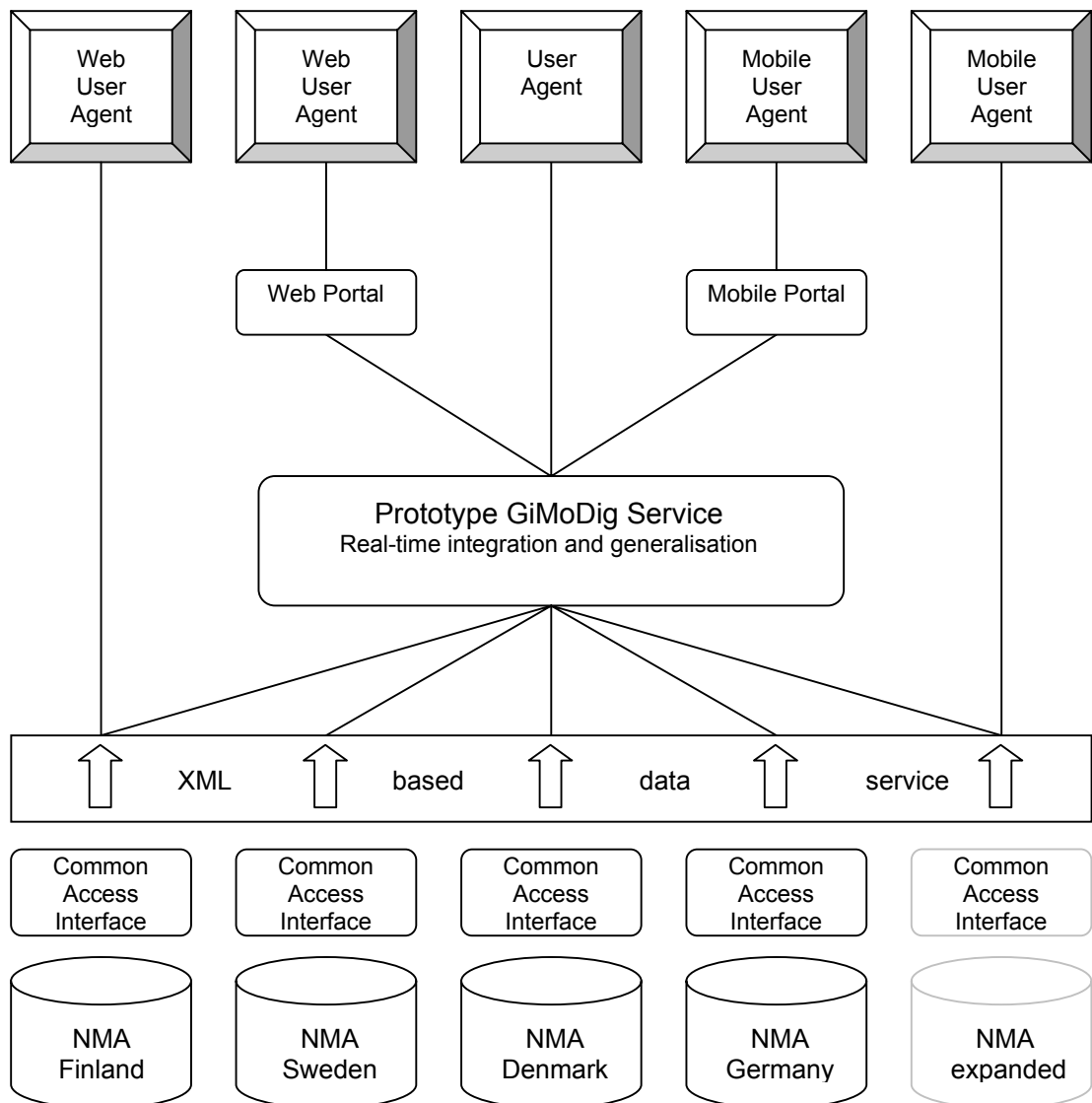


Figure 1. The Original GiMoDig System Architecture.

1.2 Layered Architecture

The layered GiMoDig architecture consists of five layers: Data Service, Data Integration, Data Processing, Portal Service and Client Application. The relevant existing OGC standard interfaces include: WFS, WMS, and OpenLS Presentation Service. Data encoding is mainly to be based on GML and SVG. The architecture, together with the most important processing modules, interfaces and the data encoding applied is shown in Figure 2. An interesting opportunity for a contribution to the standardisation activity is present in the case of the *Generalisation Service* interface, a service type for which no standardisation actions have so far been taken.

Some additional service types can be regarded as essential in the context of the GiMoDig System Architecture. These include services like Route service (route optimisation), Directory service (business locations and other Points of Interest), Geocoding service (translation from address to location) and Reverse geocoding service (translation from location to address). However, they are considered to be external to the scope of the core GiMoDig service, and therefore not further elaborated in the architecture.

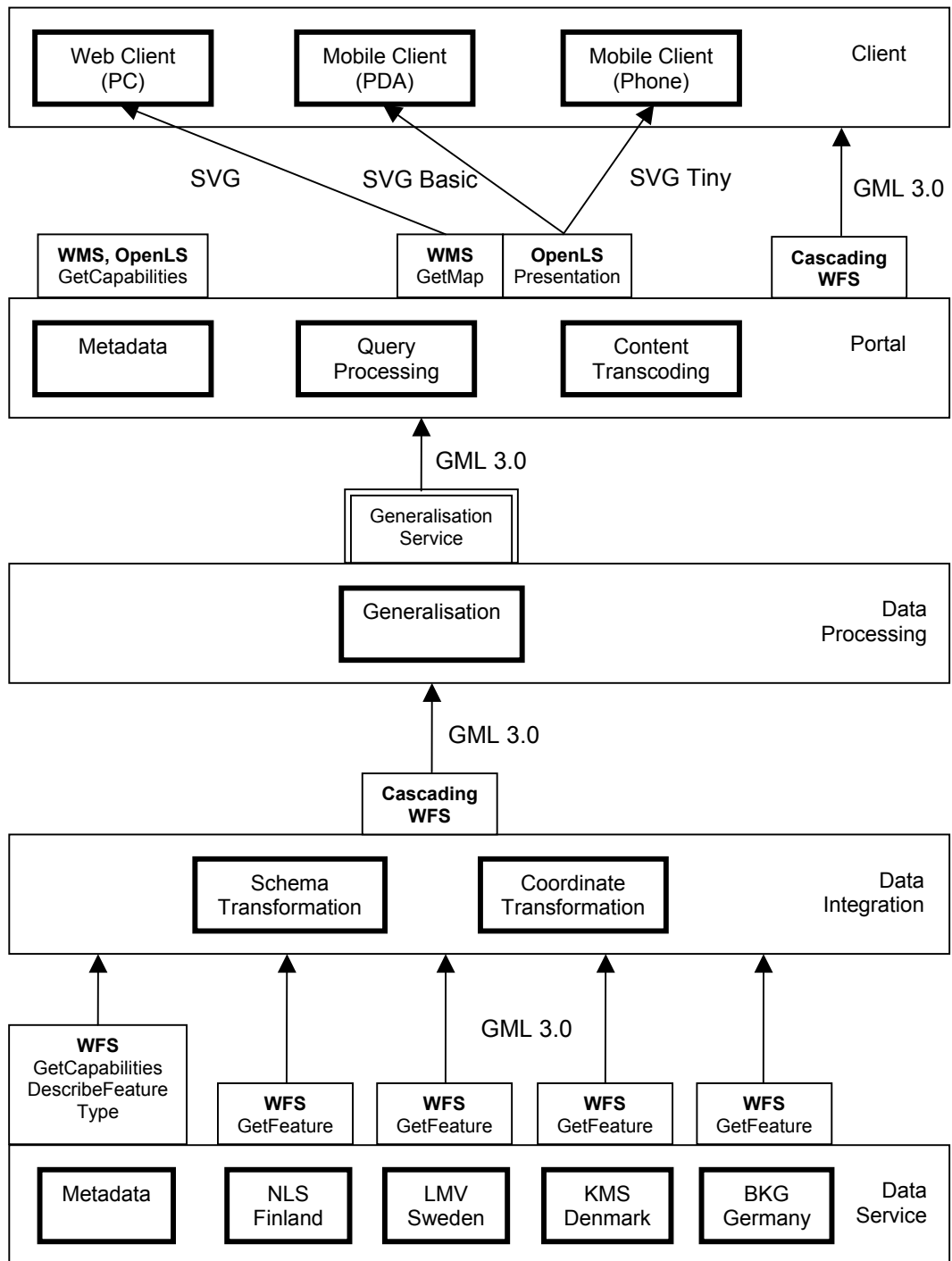


Figure 2. The five-tier GiMoDig Service Architecture.

1.3 Service Dialog

A service request initiated by a client application would initiate a rather complicated process. In the following control flow scheme the main steps involved in fulfilling the request are depicted.

Table 1. Service Dialog Scheme of the GiMoDig architecture

Requestor	Message	Control flow	Target
Client Application	<i>GetCapabilities</i> GetCapabilities document	1 → ← 2	Portal Metadata
Client Application	<i>GetMap</i> SVG image	3 → ← 17	Portal WMS, OpenLS Presentation
Portal WMS	<i>Init</i>	4 →	Content Transcoding
Portal WMS	<i>Generalise</i> Generalised GML	5 → ← 14	Generalisation Service
Generalisation Service	<i>GetFeature</i> GML data	6 → ← 13	Data Integration
Data Integration	<i>GetFeature</i> <i>GetFeature</i> GML data	7 → -----> ← 8	WFS, NMA A WFS, NMA B
Data Integration	<i>TransformSchema</i> GML, GiMoDig Schema	9 → ← 10	Schema Transformation
Data Integration	<i>TransformCoord</i> GML, ETRS89	11 → ← 12	Coordinate Transformation
Portal WMS	<i>Transcode</i> SVG image	15 → ← 16	Content Transcoding

1.4 Standardised Interfaces

WMS/WFS: GetCapabilities

One of the interfaces defined initially in the WMS specification and subsequently adopted also in the WFS and WCS specifications is the GetCapabilities interface. Being an interface through which the server is supposed to advertise its capability to provide different services, this specification forms part of the metadata and directory infrastructure of the OGC's emerging Web Services framework. In the case of the WMS the response for a GetCapabilities query consists of parameters like the spatial extent of the dataset being served, thematic map layers contained on the server and their available visualisation styles, supported co-ordinate systems, supported data formats, etc. For the WFS service the GetCapabilities request provides information about the supported schema description and feature encoding languages (presumably XML Schema and GML, respectively), available Feature Types, their spatial reference systems, spatial extents, and a possible metadata description URL address (metadata to be expressed in either ISO TC211 or FGDC format). Also a list of operations (insert, update, delete, query, lock) supported by the WFS for each of the Feature Types is to be included.

Because the GetCapabilities interface is so similar in all of the various defined OGC Web Services, the common part of this service metadata query is detailed in an OGC Discussion Paper called Basic Services Model. This document is an attempt to formulate the general framework of the OGC Web Services along the general principles set up in the ISO TC211 document ISO 19119: Geographic Information – Services. The OGC's work in this field continues in the form of the initiative: OGC Web Services, which aims at creating a common conceptual framework for interoperable spatial Web Services (OWS, 2003).

Web Feature Service, WFS

Web Feature Service (WFS) specification (version 1.0) was published as an OGC implementation specification in Sep 2002. A WFS-compliant service provides the client applications with real spatial data, geospatial Features in OGC's terminology, not with a pre-visualised map image like in a WMS-service. The resulting spatial dataset is typically expressed as an XML-encoded file. A requirement of the WFS specification is that the resulting message is encoded according to the Geography Markup Language (GML) specification, an XML vocabulary developed by the OGC for storing and distributing spatial data.

The WFS services can be divided into two separate classes. A Basic WFS is a read-only server providing only data output facilities. This kind of service supports only the GetCapabilities, DescribeFeatureType and GetFeature interfaces of the WFS specification. A Transaction WFS will let the client application also update the database (create new Features, delete and update existing Features). A service supporting Transaction WFS specification would implement Transaction interface and its defined operations Create, Delete and Update. Only the Basic WFS is discussed in the following.

The GetFeature interface enables sophisticated queries to be processed on the server. The desired set of spatial Features can be limited spatially and thematically. To be able to construct a reasonable query sentence, the client application needs to get detailed information about the Features stored in a WFS service. This information can be requested from the WFS service using the message format defined in the DescribeFeatureType interface. The result for this request is provided in the form of an XML Schema document, which describes the data model of the corresponding Feature type (attributes and their data types, geometries and their types).

The actual request for spatial data is formulated as a message complying with the Filter Encoding (FE) interface specification. The FE specification, based on the OGC's earlier Common Query Language (CQL) specification, defines the detailed format of the request parameters as an XML document. After parsing and interpreting the Filter query, the WFS service can construct the corresponding query instruction for its internal data management system. In the following example, the request asks for road network attributes "class" and "width" and geometry "centerLineOf", inside the indicated rectangle and limited to roads belonging to the class 5.

```
<GetFeature>
  <Query typeName="Road">
    <PropertyName>class</PropertyName>
    <PropertyName>width</PropertyName>
    <PropertyName>centerLineOf</PropertyName>
    <Filter>
      <And>
        <PropertyIsEqualTo>
          <PropertyName>class</PropertyName>
          <Literal>5</Literal>
        </PropertyIsEqualTo>
        <BBOX>
          <PropertyName>
            centerLineOf
          </PropertyName>
          <gml:Box
            srsName="http://www.opengis.net/gml/srs/epsg.xml#4258">
            <gml:coordinates>
              369000,7304500 370000,7306000
            </gml:coordinates>
          </gml:Box>
        </BBOX>
      </And>
    </Filter>
  </Query>
</GetFeature>
```

A WFS service enables a sophisticated spatial query to be performed on a database. In the Appendix A, a more extensive example query is presented. This query requests most of the GiMoDig Global Schema Feature types from the National Land Survey's Data Service in Finland. The query reflects the status of the GiMoDig test platform at the time of writing.

The dataset resulting from a WFS query is supposed to be returned as a GML-encoded XML-message. This geospatial dataset, expressed in real world coordinates, can be freely processed by the value-added service developers on the upper layers of the service framework to provide the desired service product. The benefits gained from the use of GML include facts like: integration of XML-encoded spatial data with other data sources becomes simple, sophisticated analysis and other processing of geospatial data on the service production layer becomes possible, and the visual layout and style of the resulting map representation can be decided as deemed appropriate. Compared with the WMS specification, the WFS provides much more flexible processing opportunities for the subsequent phases of the service chain.

WMS: GetMap, GetFeatureInfo

Web Map Service (WMS) specification (version 1.1.1) was published as an OGC implementation specification in Jan 2002. (The first version 1.0.0 was originally released in April 2000). A WMS-compliant map service provides map data for its client applications in the form of an image. Independent of the applied internal data management solutions, a WMS-service always delivers a raster or vector map image as a response to the query

message it receives. The most typical application for a WMS-service is a case where the client device is a limited-capacity terminal needing a pre-computed visualisation of the spatial data in the form of a raster image. When requested, a map server must be able to describe its services in a response to the GetCapabilities-query. The response is expressed in an XML-formatted metadata file.

The map information is requested from the WMS-service by sending a GetMap-message. The content of the map is divided into themes, or map layers. The client application must indicate which layers to include in the map image. For each map layer the server may provide a list of styles in which the corresponding layer can be visualised. The client again selects which style to use for each layer. In the GetMap-request the area of interest is indicated by four coordinate values, given in a specified coordinate system. Also the desired map coordinate system, pixel dimensions of the resulting image, and the format to be applied, can be indicated by the request parameters.

In concrete form, the WMS specification is a list of CGI parameters with well-defined semantics. The following is an example of a WMS GetMap-request which asks the service to produce road network information, visualised as a 240x320 pixel SVG image. The area of the request is indicated with ETRS89 coordinates.

```
http://gimodig.fgi.fi/restricted/bvc/WMS?  
VERSION=1.1.0&REQUEST=GetMap&  
SRS=EPSG:4258&BBOX=369000,7304500,370000,7306000&  
LAYERS=Road&STYLES=default&  
WIDTH=240&HEIGHT=320&FORMAT=image/svg+xml
```

The WMS specification defines additional user interaction functionality, named GetFeatureInfo, to the map visualisation process. It specifies a simple mechanism by which the user may click with a mouse on the map image and get, as a result, detailed information about the spatial object that was clicked on. This functionality is defined as optional in the WMS specification.

OpenLS Presentation Service

The OGC has also carried out a standardisation initiative concentrating on the needs of mobile applications needing spatial data. The process is called the Open Location Services Initiative (OpenLS). The OpenLS work will yield specifications advancing the development of interoperable mobile location services. The base document of the OpenLS initiative divides the services into three categories: Location Application Services, Gateway Services and the services already defined by the OGC. Examples of the Application Services include services like Yellow Pages, route optimisation, map display and map interaction. As possible Gateway Services, the document mentions device location service, content transcoder service and portal service. The recognised service type "device location service" is closely related to the objectives of the Location Interoperability Forum (LIF, recently merged with Open Mobile Alliance, OMA), and is to be taken directly from LIF's work to avoid inconsistencies in the resulting specifications of the two organisations.

The work of the OpenLS initiative has even more immediate effect on the evolution of mobile location services than the above-mentioned basic spatial data service specifications. Given the application-oriented approach, the developers of the various mobile services are also affected. It remains to be seen which service types actually get published as a result of the OpenLS testbed – a process was started in September 2001 and the first public demonstration was organised in October 2002 in the USA and in November 2002 in Europe.

The first draft of the OpenLS specification concentrates on interfaces related to Directory Service, Route Optimisation Service, Geocoding Service and Presentation Service. In the Presentation Service draft the service functionality has been somewhat extended when compared with the WMS GetMap interface. The most important conceptual difference is the fact that map content is divided into two categories: the background map on one hand, and the overlaid additional information on the other. This overlay information might be points of interest, optimised route, user's own location etc. The responsibilities of an OpenLS Presentation Service thus include the task of visualising the overlay geometries, provided as part of the incoming query, on top of the background map retrieved from its own database.

The second major addition is the concept of centre point context. In the WMS the spatial extent of the map can be defined only by giving the bounding box of the area requested. In the OpenLS Presentation Service specification the extent can also be defined by giving the centre point, scale of the map, and pixel size of the screen in use (Dots Per Inch, DPI). As such the specification provides some basic support for taking the user conditions into account while preparing the map. Also things like use of a clipping area and URL references to the map data are included.

1.5 Data Encoding

Geography Markup Language, GML

The basis for the GML specification rests on the OGC's work concentrating on object-oriented modelling of geospatial data. The first implementation specification of the OGC, the Simple Features (SF, 1999) Specification adapted in 1998, defines a simple practical realisation of a spatial Feature, a digital representation of a real world geographical object, schematically specified in the OGC's abstract reference model. An OGC Simple Feature consists of a set of Properties, some of which are spatial and the rest ordinary scalar attributes. It is worth noting that in the SF model a single Feature can contain multiple geometries. The SF model does not provide any support for an explicit representation of topology.

A set of spatial objects forms an entity called FeatureCollection, which is also a Feature. This arrangement enables a hierarchical structure in which an individual spatial object can be represented as a collection of other objects. The geometry in the SF model is two-dimensional and linearly interpolated. The principal geometry types are Point, LineString and Polygon. The Polygon model supports internal holes. The specification also recognises homogeneous multi-geometries consisting of several individual geometry entities (MultiPoint, MultiLineString and MultiPolygon) and a heterogeneous set of varying geometries (GeometryCollection). In the following figure the OGC's abstract Feature is depicted.

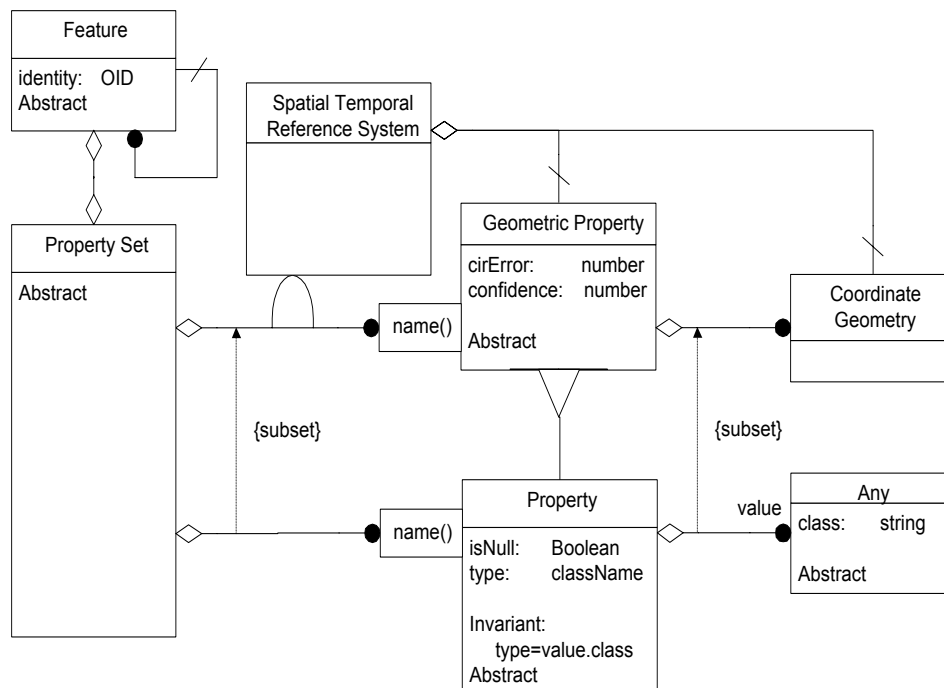


Figure 3. The abstract model of the OGC's spatial Feature.

The latest version of the GML specification was published as release 3.0 in January 2003. In the current form the specification was first adopted in Feb 2001 (version 2.0). The specification defines an XML-based encoding vocabulary for SF-compliant spatial data. The use of XML Schema as the modelling language since GML 2.0 enables the use of named types, instead of the former mechanism in which the XML element and attribute names were fixed (XML Schema, 2001). The XML Schema-based data model also facilitates the use of a genuine object-oriented approach while designing local Application Schemas. The GML specification defines an abstract data type, called *AbstractFeatureType* that can be inherited when developing concrete application level types. The defined abstract spatial Feature is actually a very simple framework in which the local application properties can be added. A GML-compliant data model has to implement the geometry level data structure, as specified in detail in the GML specification. The following is an XML Schema-based definition of the OGC's abstract Feature type. In Appendix B a preliminary version of the GiMoDig Global Schema as a GML Application Schema is presented.

```
<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>
```

The GML specification expands the original SF model in at least two respects: complex, structured data types are supported also in the case of non-spatial properties of the Feature and, in the version 2.0, geometry of the Feature could already be expressed by three coordinate values. A much more sophisticated geometry model is adopted in GML release 3.0.

The role of GML in the development of mobile services is currently clearly constrained to the server side solutions. Mobile terminals won't presumably support handling of GML on the client side in the near future. A wide implementation of the GML specification could, however, significantly facilitate development of services in which there is a need for further processing of spatial data on the application service layer of the architecture. GML-encoded spatial data will be exceptionally easy to integrate with processes that already apply XML-based data processing techniques. Also, the integration of datasets coming from several different data sources is simplified when data is available uniformly encoded, and accessible from each spatial data service through a common query interface (WFS).

Stylesheet Based Visualisation

As a result of the data processing there might be a need to express the resulting dataset as a visual representation. In the GML there is a mechanism available for indicating the default visualisation parameters of the dataset. A dedicated client application could thus process the GML-encoded dataset into a visual map display, but the solutions will then be vendor or even application-specific. However, the XML technology already provides a well-established tool for data visualisation – the stylesheet mechanism.

The widely adopted Cascading Stylesheet (CSS) technology could be applied to attach visualisation parameters to GML elements, but such a solution would not be generic. The Extensible Stylesheet Language (XSL) mechanism, and specifically its transformations-related part, XSL Transformation (XSLT, 1999), provides a flexible, standardised solution for spatial data visualisation. Using the XSLT technology, a transformation can be defined from one XML vocabulary into another XML vocabulary. This way a GML-encoded dataset can be transformed e.g. into a Scalable Vector Graphics (SVG) image. The transformation would be done by defining the transformation in an XML-file, delivering this transformation declaration, together with the source document (GML), to an XSLT processor that would then proceed to process the source document and produce the transformed document (SVG) as a result.

Scalable Vector Graphics, SVG

The SVG specification is an XML technology, developed by the World Wide Web Consortium (W3C) with the purpose of creating a standardised vector graphics format for the Web environment. The specification has reached the status of a W3C Recommendation (September 2001). The SVG format is widely seen as the obvious future standard for all kinds of vector graphics on the Web. In the texts dealing with the SVG technology, maps are frequently mentioned as a typical application area for the SVG format. Currently SVG images can be visualised in a Web browser, e.g. by using an Adobe provided SVG Viewer plugin.

For a limited capacity terminal the vector map will still be transformed into a raster image, for instance by a portal service. More powerful terminals will be able to display SVG images, thus providing much better interactivity properties for the client application. The recent work by the W3C, carried out under the title SVG Mobile to develop light-weight profiles of the SVG specification, named SVG Basic and SVG Tiny, will hopefully facilitate the use of SVG images in less powerful terminals (SVG Mobile, 2002). SVG Mobile capable viewer applications for PDAs and Java phones have actually already started appearing on the market.

2 Data Service Layer

2.1 Overview

The Data Service Layer is to be implemented in each of the participating NMAs. To facilitate data access, a standardised query interface, Web Feature Service (WFS) is to be used. There are some open source projects working to implement the specification; and a few commercial GIS vendors are also developing support for the WFS in their database products. The internal structure of a WFS node is shown in the figure below.

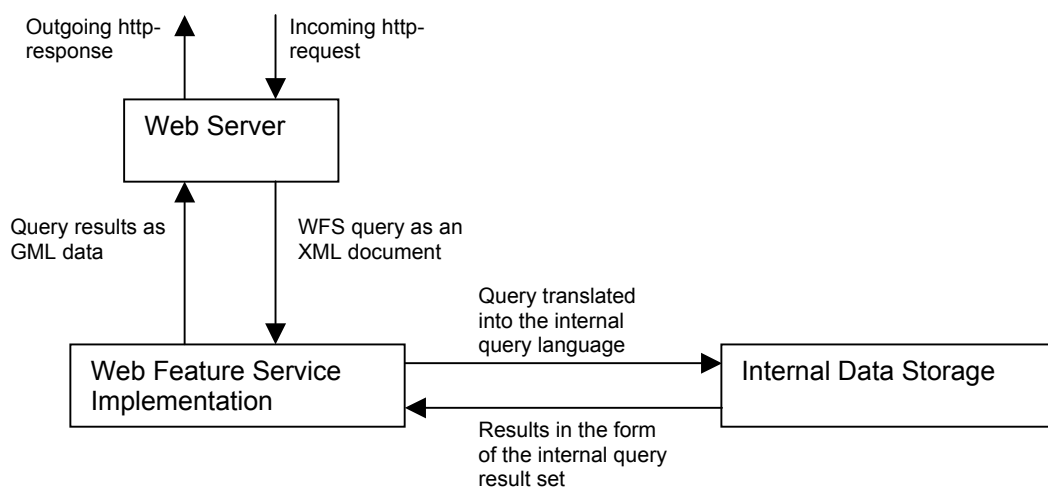


Figure 4. Internal Schema of a WFS Node.

2.2 WFS Implementation

In the case of the GiMoDig data service, only the query interfaces of the WFS will be implemented (Basic WFS). These include GetCapabilities, DescribeFeatureType and GetFeature interfaces. In addition to local service nodes, a combined GetCapabilities service of the GiMoDig pilot is to be implemented on the Portal Layer. The common GiMoDig data model (developed in the WP 5, Global Schema) will be available via the DescribeFeatureType interface. The first draft of the GiMoDig FeatureCollection, shown as a graphic representation of an XML Schema document, is presented in the following series of figures. The Road and Building Feature models are included as examples of an individual Feature class model. A complete listing of the current GiMoDig XML Schema can be found in Appendix B.

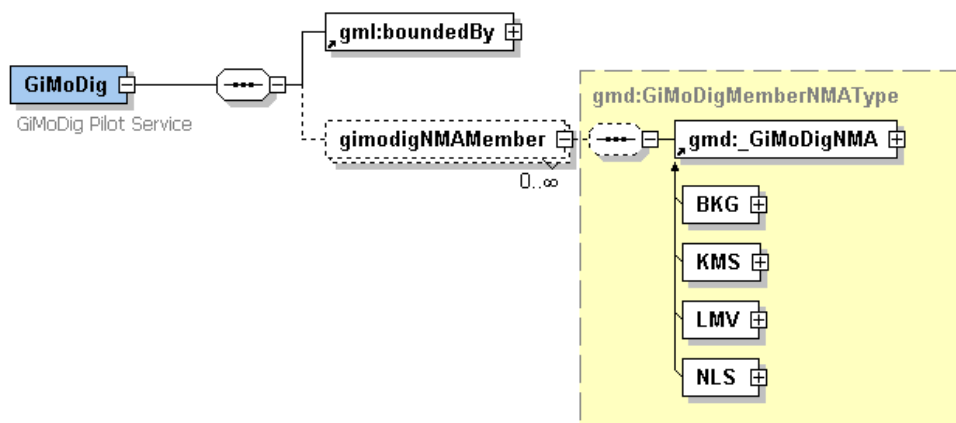


Figure 5. The main GiMoDig FeatureCollection contains individual NMAs as its member properties.

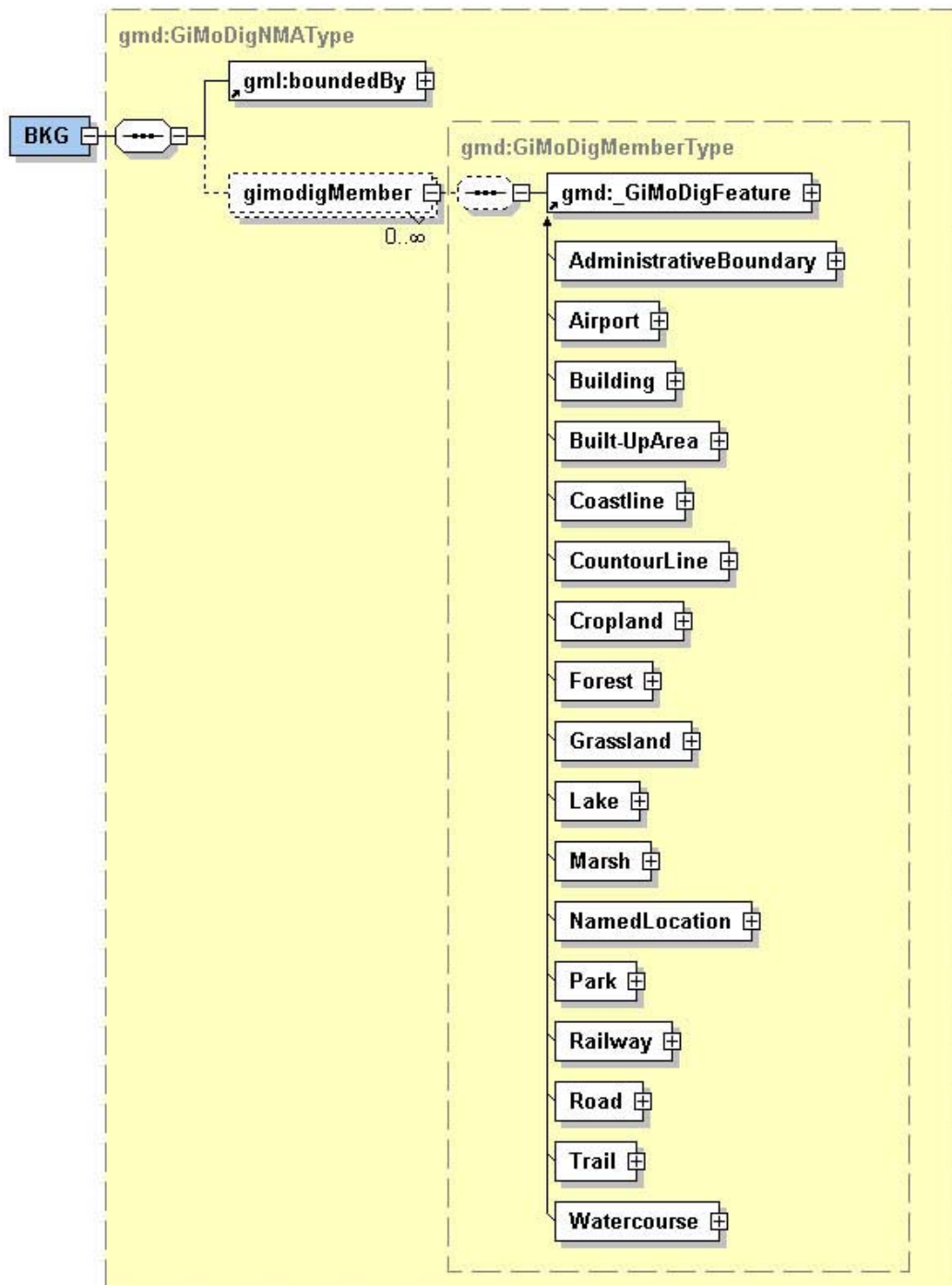


Figure 6. The current list of GiMoDig Features, presented here in the case of one NMA (Federal Agency for Cartography and Geodesy, BKG).

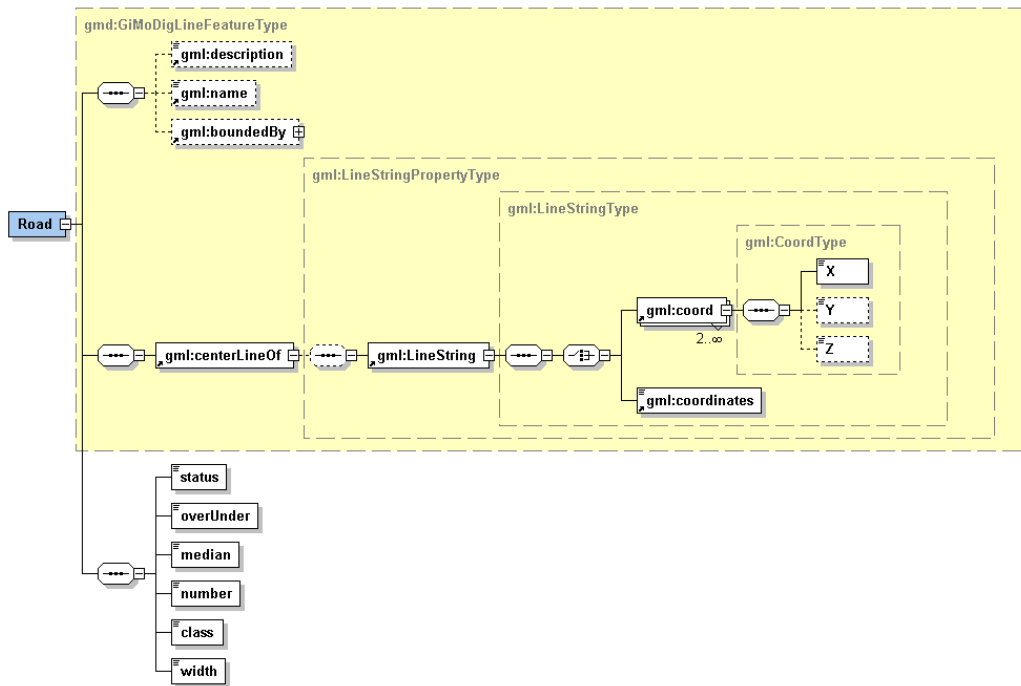


Figure 7. The Feature model of Road Feature class. Road inherits from a generic GiMoDigLineFeatureType.

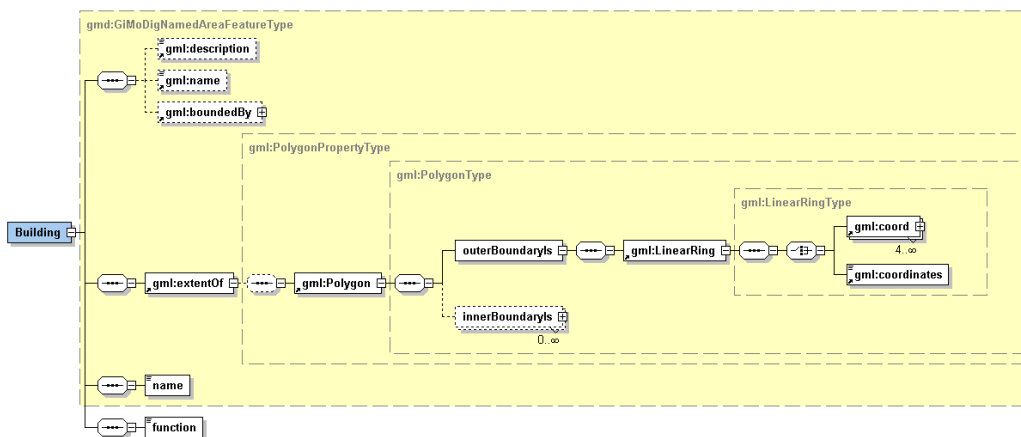


Figure 8. The Feature model of Building Feature class. Building inherits from a generic GiMoDigNamedAreaFeatureType.

The above XML Schema defines the container elements for the GiMoDig data model resulting from the harmonisation work done in WP 5. The namespace label for the data model is "http://gimodig.fgi.fi". In some cases an individual GML document might contain

data items coming from several NMAs. For this reason, under the root FeatureCollection element (GiMoDig), a second level of hierarchy is introduced (GiMoDigNMA) to distinguish between data portions originating from different NMAs. On the third level the actual GiMoDig Features will be located – Road and Building Features in the Schema are given as examples. The element structure is depicted graphically in the following figure (not based on any formal notation).

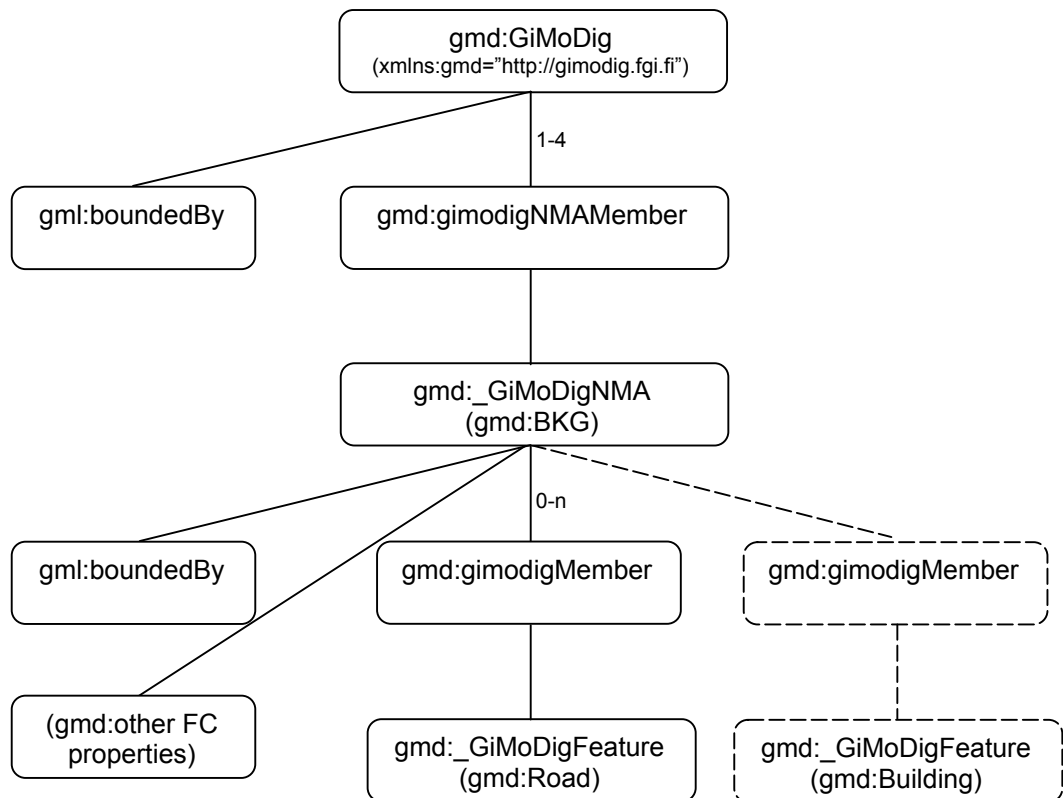


Figure 9. Graphical representation of the element structure of the GiMoDig Schema FeatureCollections.

3 Data Integration Layer

3.1 Overview

On the Data Integration layer one of the core functions of the GiMoDig project, data integration, is to be implemented (Lehto and Sarjakoski, 2003). Data integration is represented by the Schema Transformation and Coordinate Transformation modules. Although there is an open interface specification available from the OGC for the Coordinate Transformation service, and one is evidently needed in the case of the Schema Transformation, these interfaces are not planned to be specified or implemented in the GiMoDig project.

3.2 Detailed Description

Once the GetFeature request conforming to the GiMoDig Global Schema is received through the WFS interface, all the relevant NMA-specific GetFeature requests will be formulated. Then the appropriate queries are to be sent to the WFS nodes involved. On receiving the GML data from a WFS server, the service would proceed to carry out the data integration task at hand.

Data integration is conceptually divided into two phases: Coordinate Transformation and Schema Transformation. As the GiMoDig system architecture is essentially based on XML data processing, these transformations are to be carried out utilising XML tools. The XSLT process naturally fits into the task of Schema transformation. The process is declarative and transformations from individual local data models into the GiMoDig Global Schema can be defined as country-specific XSLT files. As an example, the first preliminary XSLT script for the Finnish dataset is presented in Appendix C.

During the Schema Transformation the Coordinate Transformation can be applied as an XSLT Java extension function. The workflow of the coordinate transformation is shown in the Figure 10. The parameters for the Coordinate Transformation are to be taken from the 'European Coordinate Reference Systems' directory maintained by the Federal Agency for Cartography and Geodesy (BKG), (CRS, 2001).

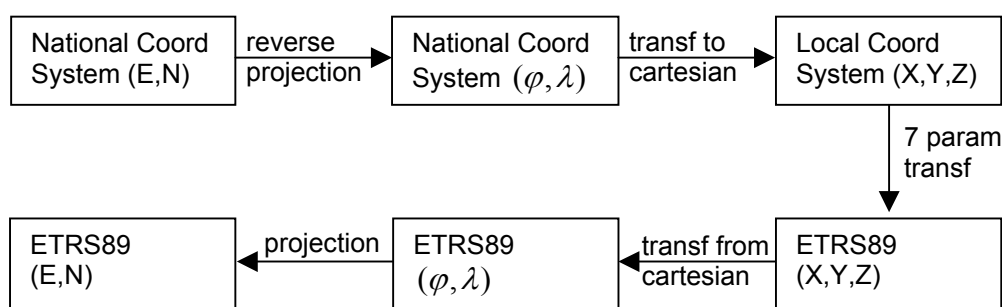


Figure 10. The Coordinate Transformation workflow.

Two exceptions to the previous schema should be mentioned. Firstly, in some cases more sophisticated computations could be applied to compensate the existing deformations in the local system. For instance, in Finland a triangle-based interpolation mechanism has been developed for that purpose. Secondly, some countries, like Denmark in the GiMoDig context, are already starting to provide their datasets in ETRS89-based coordinate systems. Representing the data as ETRS89 (φ, λ) coordinates is the most favoured approach for operational systems. In that case the Data service would serve (φ, λ) –coordinates and the Data Integration layer would only compute the required projection.

4 Data Processing Layer

4.1 Overview

Above the Data Integration Layer in the GiMoDig service architecture is the Data Processing Layer. This layer can be conceptually seen as responsible for various kinds of spatial data processing tasks that can be carried out before the actual map visualisation. In the case of the GiMoDig project, the main task to be considered on this layer is the data generalisation. Based on knowledge of the end user goal, and the mobile device in use, an appropriate generalisation process can be selected.

In the OGC's work concerning the general service architecture (OGC Web Services initiative, OWS) a service categorisation has been drafted. In that classification, along with Coordinate Transformation and Representation Conversion services, also Feature Generalisation is listed as one service category. However, no action in the OGC has so far been taken to formulate any kind of interface specifications for a generalisation service.

4.2 Internal Structure

The query comes to the service via the Generalisation Interface, to be designed later on in the project. This interface must accommodate information concerning, for instance, the spatial dataset to be requested and the level of detail (LOD) of the resulting map representation, possibly with an indication of the purpose of use and the values of the most essential user preference parameters. Also some main characteristics of the map styling to be applied must be incorporated into the Generalisation query.

There are at least two basic technical alternatives for carrying out the generalisation task. In a simple case the GML data is processed directly, for example by applying a series of XSLT transformations, to perform most simple generalisation operations like selection and simplification, together with other processes that work with a single object at a time. In a more challenging case the GML data is first uploaded to a JTS Topology Suite platform (JTS, 2003), a set of more sophisticated generalisation operators, requiring modelling interactions between cartographic objects, are applied, and the results are written out again back to the GML format. In the subsequent processing, additional XSLT transformations might still be carried out. In each of the cases the generalisation result is represented as GML data. The process is illustrated in Figure 11.

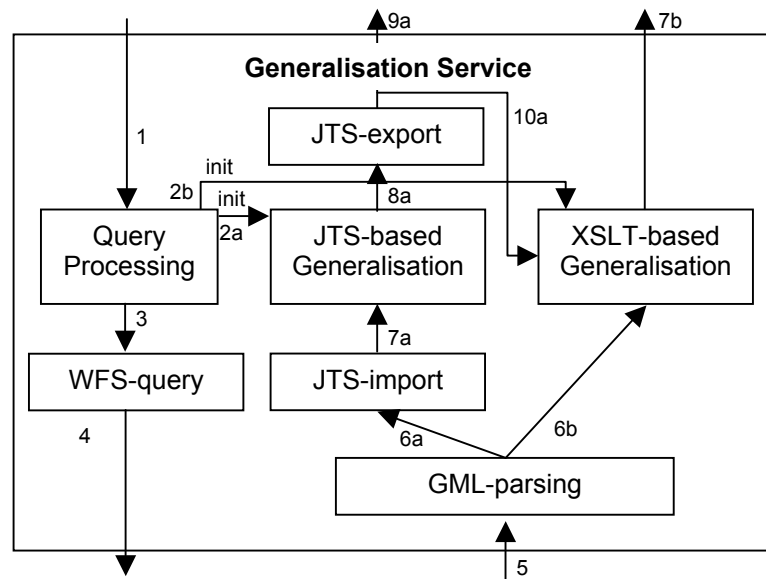


Figure 11. Generalisation Service, internal workflow. The numbers show the order in which the process is carried out. The branches a and b indicate alternative routes through the process.

JTS is a new class library in Java and it was chosen for the following reasons: Like GML, JTS conforms to the *Simple Features Specification for SQL* (standard from OGC) and it contains robust implementations of the most fundamental spatial algorithms (in 2D). Furthermore, JTS is an open source and free to use and modify in research. Computations that are planned to be carried out using JTS in the GiMoDig project include aggregation, displacement and integration of cartographic data (and possibly 3rd party data) from different sources (Harrie and Johansson, 2003).

5 Portal Service Layer

5.1 Overview

The Portal Service layer plays a two-fold role in the GiMoDig system architecture. As seen from the client's point of view the Portal Service provides the metadata query interfaces that would support the dynamic configuration of client applications. The Portal Service will also take care of the immediate communication of the visual map information by providing implementations of the essential portrayal interfaces, the WMS and the OpenLS Presentation Service. The Portal Service constitutes the main access point to the GiMoDig service platform and should thus provide some basic documentation and user instructions in the form of a professionally managed Web site.

As seen from the Applications Service layer's point of view the Portal Service works as a separate abstraction layer that insulates the core tasks, data integration and generalisation, from the specifics related to the support of various different client

platforms, communication protocols and user interaction details. The separation of these two tasks, the GiMoDig Service's main functionality and the client application support, is important for the scalability of the service. Increased modularity also makes the maintenance of the application easier. In the technical implementation sense these tasks might be integrated and run on one single server.

5.2 Detailed Structure

The two main tasks of the Portal Service layer are: 1. provision of the metadata, and 2. support for different client platforms. In the case of the GiMoDig pilot service, the former task involves implementing the metadata interface (GetCapabilities) defined in the WMS and the OpenLS Presentation Service specifications. The current work of the OGC in the area of general service architecture (OGC Web Services, OWS) may produce valuable input for this part of the work later on in the GiMoDig project. This might include describing the GiMoDig service in terms of the Web Services Description Language (WSDL, 2001) and supporting additional interfaces and new metadata schemata to be defined in subsequent OWS specifications.

The latter task consists of functionalities like map content transcoding (providing the visualised map in a format supported by the client application), user interface generation (e.g. creating the needed XHTML or SVG code or configuring other user interface components) and mechanisms for adjusting the communication details to the actual network capacity. In addition to these two main processing functionalities, the Portal Server would provide also the main access point to the system, the GiMoDig Home. The Home consists of a set of Web pages introducing the service and providing basic user guidance. These pages have to be designed separately for each supported client platform and might be periodically updated (or dynamically constructed) by appropriate queries to the Metadata service. The following figure depicts the internals of the Portal Service layer in the case of the GiMoDig system architecture.

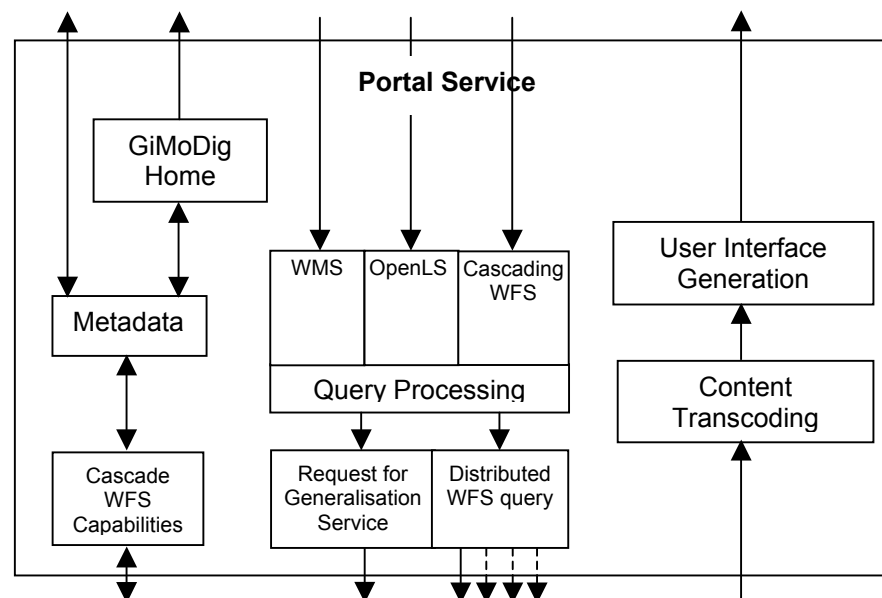


Figure 12. Internal structure of the Portal Service layer

One central question remains open concerning the use of standard interfaces, the WMS, and the OpenLS Presentation Service, in the GiMoDig service architecture: How is the client going to indicate the query specifics related to the concept of an adaptive map (Sarjakoski and Nivala, 2003)? User location can possibly be assumed to be in the centre of the requested map (Harrie et al, 2002a,b). But how to inform the service about other context parameters like user preferences, GiMoDig application to be used, etc. (Nivala and Sarjakoski, 2003)?

A partial solution is to make use of the STYLES parameter. This WMS-parameter gives a comma-separated list of pre-defined style names that correspond to the items in the same kind of list in the value of the LAYERS parameter of the query. OpenLS Presentation Service follows the same kind of approach. These style names could be possibly used in the GiMoDig service to pass over some context information. The second possibility is to make use of the Vendor-Specific Parameters (VSPs) allowed in the WMS specification. The specification dictates that servers must not require the presence of VSPs in the query, so reasonable defaults have to be defined.

6 Client Layer

On the client layer there are a few widely varying platforms to be considered. In the simple case of the traditional Web browsing PC environment, the client user interface is delivered as an XHTML document from the Portal Service. All processing related to the visualisation and user interaction is the responsibility of the browser software. Some local functionality might be added in the form of JavaScripts, but this would decrease the portability of the user interface. As an important extension to the standard Web browser functions, support for SVG based visualisation is a requirement for the Client software.

In the case of a Java client the situation is completely different. A lot of GiMoDig-specific processing could be introduced to the Client layer. However, given the limited capacity platforms that the GiMoDig project is targeting, local computations evidently have to be kept to a modest level. The following figure presents a schematic view of the possible main functions in a Java based GiMoDig client.

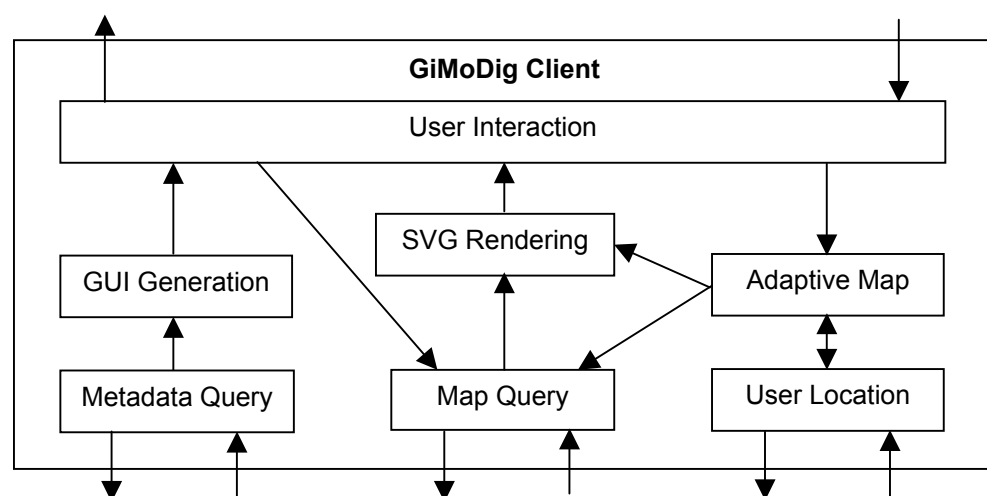


Figure 13. A Schematic View of a Java based GiMoDig client

The Client will perform a metadata query to the GiMoDig service and configure the GUI of the application, based on the query results. The User Interaction module would interface the application to the user. Certain user action could initiate a map query. The Map Query module would need to interpret the existing user profile data, the user input and the possibly available context information (like user location) to formulate an appropriate query sentence. The map image resulting from the query would finally be displayed by the SVG Rendering module.

One of the interesting functionalities that might be developed for a GiMoDig client is related to the concept of an adaptive map. Depending on a change in the user location or in some other context parameter, the map contents and visualisation characteristics might change. The Adaptive Map module would take care of this function, possibly requesting a new generalised map image from the GiMoDig service as part of the process. The concept of adaptive maps will be further considered in the WP 7 (Real-time generalisation) and the findings reported in the respective documents.

Bibliography and references

- CRS, 2001. European Coordinate Reference Systems, <http://crs.ifag.de> (accessed May 21, 2003).
- GiMoDig, 2003. Geospatial info-mobility service by real-time data-integration and generalisation, IST-project No. IST-2000-30090, at <http://gimodig.fgi.fi> (accessed May 21, 2003).
- GML, 2003. Geography Markup Language Implementation Specification, Version 3.0, at <http://www.opengis.org/techno/documents/02-023r4.pdf> (accessed May 21, 2003).
- Harrie L. and M. Johansson, 2003. Real-time data generalisation and integration using Java. *Geoforum Perspectiv*, February, 2003, pp. 29-34.
- Harrie, L., Sarjakoski, L. T. and L. Lehto, 2002a. A variable-scale map for small-display cartography. *Proceedings of the Joint International Symposium on "GeoSpatial Theory, Processing and Applications" (ISPRS/Commission IV, SDH2002)*, Ottawa, Canada, July 8-12, 2002, 6 p, CD-rom.
- Harrie, L., Sarjakoski, L. T. and L. Lehto, 2002b. A Mapping Function for Variable-Scale Maps in Small-Display Cartography. *Journal of Geospatial Engineering*, 4(2):111-123.
- Java MIDP, 2003. Mobile Information Device Profile, at <http://java.sun.com/products/midp/> (accessed May 21, 2003).
- JTS, 2003. JTS Topology Suite, at <http://www.vividsolutions.com/jts/jtshome.htm> (accessed May 21, 2003).
- Lehto L. and T. Sarjakoski, 2003. Real-time Integration of XML-Encoded Spatial Data for Mobile Use. *Proceedings of the 6th AGILE Conference on Geographic Information Science*, Lyon, France, April 24-26, pp. 97-101.
- Nissen, F., Hvas, A., Münster-Swendsen, J. and L. Brodersen, 2003. Small-Display Cartography. GiMoDig-project, IST-2000-30090, Deliverable D3.1.1, Public EC report, 66 p.
- Nivala, A.-M. and L. T. Sarjakoski, 2003. Need for Context-Aware Topographic Maps in Mobile Devices. To be published in: *Proceedings of the 9th Scandinavian Research Conference on Geographic Information Science ScanGIS'2003*, June 4-6, Espoo, Finland, 12 p.
- OGC, 2003. Open GIS Consortium Home Page, at <http://www.opengis.org> (accessed May 21, 2003).
- OpenLS, 2003. Open Location Services Initiative, at <http://www.openls.org> (accessed May 21, 2003).
- OWS, 2003. OGC Web Services Initiative, at <http://ip.opengis.org/ows2> (accessed May 21, 2003).
- Sarjakoski, L. T. and A.-M. Nivala, 2003. Context-aware maps in mobile devices. In: Salovaara, A., Kuoppala, H. and M. Nieminen, (eds.), *Perspectives on intelligent user interfaces*, Helsinki University of Technology Software Business and Engineering Institute Technical Reports 1, HUT-SoberIT-C1, Espoo, pp. 112-133. An electronic version available at <http://www.soberit.hut.fi/publications/ReportSeries/index.html> (accessed May 21, 2003).
- Sarjakoski T. and L. Lehto, 2003. Mobile Map Services Based on Open System Architecture. To be published in: *Proceedings of the 21st International Cartographic Conference*, August 10-16, Durban, South Africa.

Sarjakoski, T., Sarjakoski, L. T., Lehto, L., Sester, M., Illert, A., Nissen, F., Rystedt, R. and R. Ruotsalainen, 2002. Geospatial Info-mobility Services - A Challenge for National Mapping Agencies. Proceedings of the Joint International Symposium on "GeoSpatial Theory, Processing and Applications" (ISPRS/Commission IV, SDH2002), Ottawa, Canada, July 8-12, 2002, 5 p, CD-rom.

SF, 1999. Simple Features Specification, at <http://www.opengis.org/techno/implementation.htm> (accessed May 21, 2003).

SVG, 2001. Scalable Vector Graphics 1.0 Specification, at <http://www.w3.org/TR/SVG> (accessed May 21, 2003).

SVG Mobile, 2002. Mobile SVG Profiles: SVG Tiny and SVG Basic, at <http://www.w3.org/TR/SVGMobile/> (accessed May 21, 2003).

WFS, 2002. Web Feature Server Implementation Specification, Version 1.0.0, at <http://www.opengis.org/techno/specs/02-058.pdf> (accessed May 21, 2003).

WMS, 2002. Web Map Service Implementation Specification, Version 1.1.1, at <http://www.opengis.org/techno/specs/01-068r3.pdf> (accessed May 21, 2003).

WSDL, 2001. Web Services Description Language 1.2, at <http://www.w3.org/TR/wsdl12/> (accessed May 21, 2003).

XML, 2000. Extensible Markup Language 1.0 (Second Edition), at <http://www.w3.org/TR/2000/REC-xml-20001006> (accessed May 21, 2003).

XML Schema, 2001. XML Schema Part 0,1 and 2, at <http://www.w3.org/XML/Schema#dev> (accessed May 21, 2003).

XSLT, 1999. XSL Transformations, Version 1.0, at <http://www.w3.org/TR/xslt> (accessed May 21, 2003).

Appendix A

WFS Query, NLS

This Web Feature Service (WFS) query requests most of the GiMoDig Global Schema Feature types from the Finnish National Land Survey's (NLS) Data Service. The query reflects the status of the GiMoDig test platform at the time of writing.

```
<?xml version="1.0" encoding="UTF-8"?>
<GetFeature outputFormat="GML2" handle="geoserverQuery">
  <Query handle="query00" typeName="hallinnollinen_jaotus_viiva" version="1.0">
    <PropertyName>the_geom</PropertyName>
    <PropertyName>luokka</PropertyName>
    <Filter xmlns:gml="http://www.opengis.net/gml">
      <And>
        <BBOX>
          <PropertyName>the_geom</PropertyName>
          <gml:Box srsName="EPSG:2393">
            <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
          </gml:Box>
        </BBOX>
      </And>
    </Filter>
  </Query>
  <Query handle="query10" typeName="maasto1_polygoni" version="1.0">
    <PropertyName>the_geom</PropertyName>
    <PropertyName>luokka</PropertyName>
    <Filter xmlns:gml="http://www.opengis.net/gml">
      <And>
        <BBOX>
          <PropertyName>the_geom</PropertyName>
          <gml:Box srsName="EPSG:2393">
            <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
          </gml:Box>
        </BBOX>
        <PropertyIsEqualTo>
          <PropertyName>luokka</PropertyName>
          <Literal>36313</Literal>
        </PropertyIsEqualTo>
      </And>
    </Filter>
  </Query>
  <Query handle="query11" typeName="maasto1_viiva" version="1.0">
    <PropertyName>the_geom</PropertyName>
    <PropertyName>luokka</PropertyName>
    <Filter xmlns:gml="http://www.opengis.net/gml">
      <And>
        <BBOX>
          <PropertyName>the_geom</PropertyName>
          <gml:Box srsName="EPSG:2393">
            <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
          </gml:Box>
        </BBOX>
        <PropertyIsEqualTo>
          <PropertyName>luokka</PropertyName>
          <Literal>36312</Literal>
        </PropertyIsEqualTo>
      </And>
    </Filter>
  </Query>
  <Query handle="query20" typeName="maasto1_polygoni" version="1.0">
```

```

<PropertyName>the_geom</PropertyName>
<PropertyName>luokka</PropertyName>
<Filter xmlns:gml="http://www.opengis.net/gml">
  <And>
    <BBOX>
      <PropertyName>the_geom</PropertyName>
      <gml:Box srsName="EPSG:2393">
        <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
      </gml:Box>
    </BBOX>
    <PropertyIsEqualTo>
      <PropertyName>luokka</PropertyName>
      <Literal>36200</Literal>
    </PropertyIsEqualTo>
  </And>
</Filter>
</Query>
<Query handle="query30" typeName="maasto1_polygони" version="1.0">
  <PropertyName>the_geom</PropertyName>
  <PropertyName>luokka</PropertyName>
  <Filter xmlns:gml="http://www.opengis.net/gml">
    <And>
      <BBOX>
        <PropertyName>the_geom</PropertyName>
        <gml:Box srsName="EPSG:2393">
          <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
        </gml:Box>
      </BBOX>
      <Or>
        <PropertyIsEqualTo>
          <PropertyName>luokka</PropertyName>
          <Literal>35411</Literal>
        </PropertyIsEqualTo>
        <PropertyIsEqualTo>
          <PropertyName>luokka</PropertyName>
          <Literal>35412</Literal>
        </PropertyIsEqualTo>
        <PropertyIsEqualTo>
          <PropertyName>luokka</PropertyName>
          <Literal>35421</Literal>
        </PropertyIsEqualTo>
        <PropertyIsEqualTo>
          <PropertyName>luokka</PropertyName>
          <Literal>35422</Literal>
        </PropertyIsEqualTo>
      </Or>
    </And>
  </Filter>
</Query>
<Query handle="query40" typeName="maasto1_polygони" version="1.0">
  <PropertyName>the_geom</PropertyName>
  <PropertyName>luokka</PropertyName>
  <Filter xmlns:gml="http://www.opengis.net/gml">
    <And>
      <BBOX>
        <PropertyName>the_geom</PropertyName>
        <gml:Box srsName="EPSG:2393">
          <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
        </gml:Box>
      </BBOX>
      <PropertyIsEqualTo>
        <PropertyName>luokka</PropertyName>
        <Literal>32900</Literal>
      </PropertyIsEqualTo>
    </And>
  </Filter>
</Query>
<Query handle="query50" typeName="rakennus_polygони" version="1.0">
  <PropertyName>the_geom</PropertyName>
  <PropertyName>luokka</PropertyName>
  <Filter xmlns:gml="http://www.opengis.net/gml">
    <And>
      <BBOX>
        <PropertyName>the_geom</PropertyName>

```

```

        <gml:Box srsName="EPSG:2393">
          <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
        </gml:Box>
      </BBOX>
    </And>
  </Filter>
</Query>
<Query handle="query60" typeName="korkeussuhteet_viiva" version="1.0">
  <PropertyName>the_geom</PropertyName>
  <Filter xmlns:gml="http://www.opengis.net/gml">
    <And>
      <BBOX>
        <PropertyName>the_geom</PropertyName>
        <gml:Box srsName="EPSG:2393">
          <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
        </gml:Box>
      </BBOX>
      <PropertyIsEqualTo>
        <PropertyName>luokka</PropertyName>
        <Literal>52100</Literal>
      </PropertyIsEqualTo>
    </And>
  </Filter>
</Query>
<Query handle="query70" typeName="maasto1_polygoni" version="1.0">
  <PropertyName>the_geom</PropertyName>
  <PropertyName>luokka</PropertyName>
  <Filter xmlns:gml="http://www.opengis.net/gml">
    <And>
      <BBOX>
        <PropertyName>the_geom</PropertyName>
        <gml:Box srsName="EPSG:2393">
          <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
        </gml:Box>
      </BBOX>
      <PropertyIsEqualTo>
        <PropertyName>luokka</PropertyName>
        <Literal>32611</Literal>
      </PropertyIsEqualTo>
    </And>
  </Filter>
</Query>
<Query handle="query90" typeName="liikenneverkot_viiva" version="1.0">
  <PropertyName>the_geom</PropertyName>
  <PropertyName>luokka</PropertyName>
  <Filter xmlns:gml="http://www.opengis.net/gml">
    <And>
      <BBOX>
        <PropertyName>the_geom</PropertyName>
        <gml:Box srsName="EPSG:2393">
          <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
        </gml:Box>
      </BBOX>
      <Or>
        <PropertyIsEqualTo>
          <PropertyName>luokka</PropertyName>
          <Literal>14111</Literal>
        </PropertyIsEqualTo>
        <PropertyIsEqualTo>
          <PropertyName>luokka</PropertyName>
          <Literal>14112</Literal>
        </PropertyIsEqualTo>
      </Or>
    </And>
  </Filter>
</Query>
<Query handle="query100" typeName="liikenneverkot_viiva" version="1.0">
  <PropertyName>the_geom</PropertyName>
  <PropertyName>luokka</PropertyName>
  <Filter xmlns:gml="http://www.opengis.net/gml">
    <And>
      <BBOX>
        <PropertyName>the_geom</PropertyName>
        <gml:Box srsName="EPSG:2393">

```

```

        <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
    </gml:Box>
</BBOX>
<Or>
    <PropertyIsEqualTo>
        <PropertyName>luokka</PropertyName>
        <Literal>12111</Literal>
    </PropertyIsEqualTo>
    <PropertyIsEqualTo>
        <PropertyName>luokka</PropertyName>
        <Literal>12112</Literal>
    </PropertyIsEqualTo>
    <PropertyIsEqualTo>
        <PropertyName>luokka</PropertyName>
        <Literal>12121</Literal>
    </PropertyIsEqualTo>
    <PropertyIsEqualTo>
        <PropertyName>luokka</PropertyName>
        <Literal>12122</Literal>
    </PropertyIsEqualTo>
    <PropertyIsEqualTo>
        <PropertyName>luokka</PropertyName>
        <Literal>12131</Literal>
    </PropertyIsEqualTo>
    <PropertyIsEqualTo>
        <PropertyName>luokka</PropertyName>
        <Literal>12132</Literal>
    </PropertyIsEqualTo>
    <PropertyIsEqualTo>
        <PropertyName>luokka</PropertyName>
        <Literal>12141</Literal>
    </PropertyIsEqualTo>
</Or>
</And>
</Filter>
</Query>
<Query handle="query110" typeName="liikenneverkot_viiva" version="1.0">
    <PropertyName>the_geom</PropertyName>
    <PropertyName>luokka</PropertyName>
    <Filter xmlns:gml="http://www.opengis.net/gml">
        <And>
            <BBOX>
                <PropertyName>the_geom</PropertyName>
                <gml:Box srsName="EPSG:2393">
                    <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
                </gml:Box>
            </BBOX>
            <Or>
                <PropertyIsEqualTo>
                    <PropertyName>luokka</PropertyName>
                    <Literal>12313</Literal>
                </PropertyIsEqualTo>
                <PropertyIsEqualTo>
                    <PropertyName>luokka</PropertyName>
                    <Literal>12314</Literal>
                </PropertyIsEqualTo>
                <PropertyIsEqualTo>
                    <PropertyName>luokka</PropertyName>
                    <Literal>12316</Literal>
                </PropertyIsEqualTo>
            </Or>
        </And>
    </Filter>
</Query>
<Query handle="query120" typeName="maasto1_teksti" version="1.0">
    <PropertyName>the_geom</PropertyName>
    <PropertyName>luokka</PropertyName>
    <PropertyName>teksti</PropertyName>
    <PropertyName>suunta</PropertyName>
    <PropertyName>siirt_dx</PropertyName>
    <PropertyName>siirt_dy</PropertyName>
    <Filter xmlns:gml="http://www.opengis.net/gml">
        <And>
            <BBOX>

```

```

        <PropertyName>the_geom</PropertyName>
        <gml:Box srsName="EPSG:2393">
          <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
        </gml:Box>
      </BBOX>
    </And>
  </Filter>
</Query>
<Query handle="query121" typeName="rakennus_teksti" version="1.0">
  <PropertyName>the_geom</PropertyName>
  <PropertyName>luokka</PropertyName>
  <PropertyName>teksti</PropertyName>
  <PropertyName>suunta</PropertyName>
  <PropertyName>siirt_dx</PropertyName>
  <PropertyName>siirt_dy</PropertyName>
  <Filter xmlns:gml="http://www.opengis.net/gml">
    <And>
      <BBOX>
        <PropertyName>the_geom</PropertyName>
        <gml:Box srsName="EPSG:2393">
          <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
        </gml:Box>
      </BBOX>
    </And>
  </Filter>
</Query>
<Query handle="query150" typeName="maasto1_polygoni" version="1.0">
  <PropertyName>the_geom</PropertyName>
  <PropertyName>luokka</PropertyName>
  <Filter xmlns:gml="http://www.opengis.net/gml">
    <And>
      <BBOX>
        <PropertyName>the_geom</PropertyName>
        <gml:Box srsName="EPSG:2393">
          <gml:coordinates>3372954,7310557 3374288,7311557</gml:coordinates>
        </gml:Box>
      </BBOX>
      <Or>
        <PropertyIsEqualTo>
          <PropertyName>luokka</PropertyName>
          <Literal>32612</Literal>
        </PropertyIsEqualTo>
        <PropertyIsEqualTo>
          <PropertyName>luokka</PropertyName>
          <Literal>32800</Literal>
        </PropertyIsEqualTo>
      </Or>
    </And>
  </Filter>
</Query>
</GetFeature>

```

Appendix B

GiMoDig Schema

This XML Schema document is a preliminary version of the GiMoDig Global Schema, designed as a GML Application Schema.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsd:schema targetNamespace="http://gimodig.fgi.fi" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:gmd="http://gimodig.fgi.fi" elementFormDefault="qualified">
  <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="http://gimodig.fgi.fi/xsd/feature.xsd"/>
  <!-- Definitions related to the FeatureCollection hierarchy of the GiMoDig Schema -->
  <xsd:element name="_GiMoDigNMA" type="gmd:GiMoDigNMAType" abstract="true"
substitutionGroup="gml:_FeatureCollection"/>
  <xsd:element name="_GiMoDigFeature" type="gml:AbstractFeatureType" abstract="true"
substitutionGroup="gml:_Feature"/>
  <xsd:complexType name="GiMoDigMemberNMAType">
    <xsd:complexContent>
      <xsd:restriction base="gml:FeatureAssociationType">
        <xsd:sequence minOccurs="0">
          <xsd:element ref="gmd:_GiMoDigNMA"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="GiMoDig" substitutionGroup="gml:_FeatureCollection">
    <xsd:annotation>
      <xsd:documentation>GiMoDig Pilot Service</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:restriction base="gml:AbstractFeatureCollectionType">
          <xsd:sequence>
            <xsd:element ref="gml:boundedBy"/>
            <xsd:element name="gimodigNMAMember" type="gmd:GiMoDigMemberNMAType" minOccurs="0"
maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:restriction>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="GiMoDigMemberType">
    <xsd:complexContent>
      <xsd:restriction base="gml:FeatureAssociationType">
        <xsd:sequence minOccurs="0">
          <xsd:element ref="gmd:_GiMoDigFeature"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="GiMoDigNMAType">
    <xsd:complexContent>
      <xsd:restriction base="gml:AbstractFeatureCollectionType">
        <xsd:sequence>
          <xsd:element ref="gml:boundedBy"/>
          <xsd:element name="gimodigMember" type="gmd:GiMoDigMemberType" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
```

```

<!-- Definitions for individual NMAs -->
<xsd:element name="BKG" type="gmd:GiMoDigNMAType" substitutionGroup="gmd:_GiMoDigNMA"/>
<xsd:element name="KMS" type="gmd:GiMoDigNMAType" substitutionGroup="gmd:_GiMoDigNMA"/>
<xsd:element name="LMV" type="gmd:GiMoDigNMAType" substitutionGroup="gmd:_GiMoDigNMA"/>
<xsd:element name="NLS" type="gmd:GiMoDigNMAType" substitutionGroup="gmd:_GiMoDigNMA"/>
<!-- Named Types -->
<xsd:complexType name="GiMoDigLineFeatureType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element ref="gml:centerLineOf"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="GiMoDigNamedLineFeatureType">
  <xsd:complexContent>
    <xsd:extension base="gmd:GiMoDigLineFeatureType">
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="GiMoDigAreaFeatureType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element ref="gml:extentOf"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="GiMoDigNamedAreaFeatureType">
  <xsd:complexContent>
    <xsd:extension base="gmd:GiMoDigAreaFeatureType">
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="OverUnderType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="over"/>
    <xsd:enumeration value="under"/>
  </xsd:restriction>
</xsd:simpleType>
<!-- GiMoDig Features -->
<xsd:element name="Coastline" type="gmd:GiMoDigLineFeatureType" substitutionGroup="gmd:_GiMoDigFeature"/>
<xsd:element name="AdministrativeBoundary" substitutionGroup="gmd:_GiMoDigFeature">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="gmd:GiMoDigNamedLineFeatureType">
        <xsd:sequence>
          <xsd:element name="level" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Watercourse" substitutionGroup="gmd:_GiMoDigFeature">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element ref="gml:centerLineOf" minOccurs="0"/>
          <xsd:element ref="gml:extentOf" minOccurs="0"/>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="overUnder" type="gmd:OverUnderType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

```

```

</xsd:element>
<xsd:element name="Lake" type="gmd:GiMoDigNamedAreaFeatureType" substitutionGroup="gmd:_GiMoDigFeature"/>
<xsd:element name="Marsh" type="gmd:GiMoDigAreaFeatureType" substitutionGroup="gmd:_GiMoDigFeature"/>
<xsd:element name="Park" type="gmd:GiMoDigAreaFeatureType" substitutionGroup="gmd:_GiMoDigFeature"/>
<xsd:element name="Building" substitutionGroup="gmd:_GiMoDigFeature">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="gmd:GiMoDigNamedAreaFeatureType">
        <xsd:sequence>
          <xsd:element name="function" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="CountourLine" substitutionGroup="gmd:_GiMoDigFeature">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="gmd:GiMoDigLineFeatureType">
        <xsd:sequence>
          <xsd:element name="heightValue" type="xsd:positiveInteger"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Cropland" type="gmd:GiMoDigAreaFeatureType" substitutionGroup="gmd:_GiMoDigFeature"/>
<xsd:element name="NamedLocation" substitutionGroup="gmd:_GiMoDigFeature">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element ref="gml:position"/>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="category" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Built-UpArea" type="gmd:GiMoDigNamedAreaFeatureType"
substitutionGroup="gmd:_GiMoDigFeature"/>
<xsd:element name="Railway" substitutionGroup="gmd:_GiMoDigFeature">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="gmd:GiMoDigLineFeatureType">
        <xsd:sequence>
          <xsd:element name="overUnder" type="gmd:OverUnderType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Road" substitutionGroup="gmd:_GiMoDigFeature">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="gmd:GiMoDigLineFeatureType">
        <xsd:sequence>
          <xsd:element name="status" type="xsd:string"/>
          <xsd:element name="overUnder" type="gmd:OverUnderType"/>
          <xsd:element name="median" type="xsd:string"/>
          <xsd:element name="number" type="xsd:unsignedShort"/>
          <xsd:element name="class" type="xsd:unsignedShort"/>
          <xsd:element name="width" type="xsd:unsignedShort"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Trail" substitutionGroup="gmd:_GiMoDigFeature">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="gmd:GiMoDigLineFeatureType">
        <xsd:sequence>

```

```
        <xsd:element name="overUnder" type="gmd:OverUnderType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
</xsd:element>
<xsd:element name="Airport" type="gmd:GiMoDigNamedAreaFeatureType" substitutionGroup="gmd:_GiMoDigFeature"/>
<xsd:element name="Forest" type="gmd:GiMoDigAreaFeatureType" substitutionGroup="gmd:_GiMoDigFeature"/>
<xsd:element name="Grassland" type="gmd:GiMoDigAreaFeatureType" substitutionGroup="gmd:_GiMoDigFeature"/>
</xsd:schema>
```

Appendix C

XSLT Declarations – NLS to GiMoDig

The Schema Transformations are defined in the GiMoDig project as XSLT declarations. Thus, transformations from the individual local data models into the GiMoDig Global Schema can be defined as country-specific XSLT files. In this example, the first preliminary XSLT script for the Finnish dataset is shown.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
  xmlns:xalan="http://xml.apache.org/xslt"
  xmlns:gis="xalan://gis.CoordTrans"
  xmlns:gisLS="xalan://gis.SimplWeed"
  xmlns:gisNLS="xalan://gis.CoordSysTransNLS"
  xmlns:gisNLS2="xalan://gis.CoordSysTransNLS2"
  xmlns:gmd="http://gimodig.fgi.fi"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:wfs="http://www.opengis.net/wfs"
  xmlns="http://www.opengis.net/gml" exclude-result-prefixes="gis gisLS gisNLS gisNLS2 wfs">

  <xsl:template match="/wfs:FeatureCollection">
    <gmd:GiMoDig>
      <gmd:gimodigNMAMember>
        <gmd:NLS>
          <boundedBy><null>unavailable</null></boundedBy>
          <xsl:apply-templates select="gml:featureMember"/>
        </gmd:NLS>
      </gmd:gimodigNMAMember>
    </gmd:GiMoDig>
  </xsl:template>

  <xsl:template match="gml:featureMember">
    <gmd:gimodigMember>
      <xsl:apply-templates/>
    </gmd:gimodigMember>
  </xsl:template>

  <xsl:template match="liikenneverkot_viiva">
    <xsl:choose>
      <xsl:when test="luokka &lt; 12300">
        <gmd:Road fid="{@fid}">
          <xsl:apply-templates select="the_geom"/>
          <gmd:status></gmd:status>
          <gmd:overUnder>over</gmd:overUnder>
          <gmd:median></gmd:median>
          <gmd:number>0</gmd:number>
          <gmd:class><xsl:value-of select="luokka"/></gmd:class>
          <gmd:width>0</gmd:width>
        </gmd:Road>
      </xsl:when>
      <xsl:when test="luokka = 14111 or luokka = 14112">
        <gmd:Railway fid="{@fid}">
          <xsl:apply-templates select="the_geom"/>
        </gmd:Railway>
      </xsl:when>
      <xsl:when test="luokka = 12313 or luokka = 12314 or luokka = 12316">
        <gmd:Trail fid="{@fid}">
          <xsl:apply-templates select="the_geom"/>
          <gmd:overUnder>over</gmd:overUnder>
        </gmd:Trail>
      </xsl:when>
    </xsl:choose>
  </xsl:template>

```

```

        </xsl:when>
        <xsl:otherwise>
            <gmd:Unknown>
                <xsl:apply-templates select="luokka"/>
            </gmd:Unknown>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

<xsl:template match="hallinnollinen_jaotus_viiva">
    <gmd:AdministrativeBoundary fid="{@fid}">
        <xsl:apply-templates select="the_geom"/>
        <gmd:name></gmd:name>
        <xsl:choose>
            <xsl:when test="luokka = 84111">
                <gmd:level>national</gmd:level>
            </xsl:when>
            <xsl:when test="luokka = 84113">
                <gmd:level>municipal</gmd:level>
            </xsl:when>
        </xsl:choose>
    </gmd:AdministrativeBoundary>
</xsl:template>

<xsl:template match="korkeussuhteet_viiva">
    <gmd:ContourLine fid="{@fid}">
        <xsl:apply-templates select="the_geom"/>
        <gmd:heightValue/>
    </gmd:ContourLine>
</xsl:template>

<xsl:template match="maasto1_viiva">
    <xsl:choose>
        <xsl:when test="luokka = 36312">
            <gmd:Watercourse fid="{@fid}">
                <xsl:apply-templates select="the_geom"/>
                <gmd:name/>
                <gmd:overUnder>over</gmd:overUnder>
            </gmd:Watercourse>
        </xsl:when>
        <xsl:otherwise>
            <xsl:if test="kartoglk = 36200 or kartoglk = 36313">
                <gmd:Coastline fid="{@fid}">
                    <xsl:apply-templates select="the_geom"/>
                </gmd:Coastline>
            </xsl:if>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

<xsl:template match="rakennus_polygoni">
    <gmd:Building fid="{@fid}">
        <xsl:apply-templates select="the_geom"/>
        <xsl:variable name="func" select="luokka"/>
        <xsl:choose>
            <xsl:when test="$func = '42211' or $func = '42212' or $func = '42221' or $func = '42241' or $func = '42251'"><gmd:function>main</gmd:function></xsl:when>
            <xsl:when test="$func = '42231' or $func = '42261'"><gmd:function>out</gmd:function></xsl:when>
            <xsl:otherwise><gmd:function>unknown: <xsl:value-of select="$func"/></gmd:function></xsl:otherwise>
        </xsl:choose>
        <gmd:name></gmd:name>
    </gmd:Building>
</xsl:template>

<xsl:template match="maasto1_polygoni">
    <xsl:choose>
        <xsl:when test="luokka = 36313">
            <gmd:Watercourse fid="{@fid}">
                <xsl:apply-templates select="the_geom"/>
            </gmd:Watercourse>
        </xsl:when>
        <xsl:when test="luokka = 36200">
            <gmd:Lake fid="{@fid}">
                <xsl:apply-templates select="the_geom"/>
            </gmd:Lake>
        </xsl:when>
    </xsl:choose>
</xsl:template>

```

```

        </gmd:Lake>
    </xsl:when>
    <xsl:when test="luokka = 35411 or luokka = 35412 or luokka = 35421 or luokka = 35422">
        <gmd:Marsh fid="{@fid}">
            <xsl:apply-templates select="the_geom"/>
        </gmd:Marsh>
    </xsl:when>
    <xsl:when test="luokka = 32900">
        <gmd:Park fid="{@fid}">
            <xsl:apply-templates select="the_geom"/>
        </gmd:Park>
    </xsl:when>
    <xsl:when test="luokka = 32611">
        <gmd:Cropland fid="{@fid}">
            <xsl:apply-templates select="the_geom"/>
        </gmd:Cropland>
    </xsl:when>
    <xsl:when test="luokka = 32612 or luokka = 32800">
        <gmd:Grassland fid="{@fid}">
            <xsl:apply-templates select="the_geom"/>
        </gmd:Grassland>
    </xsl:when>
</xsl:choose>
</xsl:template>

<xsl:template match="maasto1_teksti">
    <gmd:NamedLocation fid="{@fid}">
        <xsl:apply-templates select="the_geom"/>
        <gmd:rotation><xsl:value-of select="suunta div -174.533"/></gmd:rotation>
        <gmd:category>terrain</gmd:category>
        <gmd:name><xsl:value-of select="teksti"/></gmd:name>
    </gmd:NamedLocation>
</xsl:template>

<xsl:template match="rakennus_teksti">
    <gmd:NamedLocation fid="{@fid}">
        <xsl:apply-templates select="the_geom"/>
        <gmd:rotation><xsl:value-of select="suunta div -174.533"/></gmd:rotation>
        <xsl:variable name="categ" select="luokka"/>
        <xsl:choose>
            <xsl:when test="$categ = 48111">
                <gmd:category>city</gmd:category>
            </xsl:when>
            <xsl:when test="$categ = 48120">
                <gmd:category>neighbourhood</gmd:category>
            </xsl:when>
            <xsl:otherwise>
                <gmd:category>other</gmd:category>
            </xsl:otherwise>
        </xsl:choose>
        <gmd:name><xsl:value-of select="teksti"/></gmd:name>
    </gmd:NamedLocation>
</xsl:template>

<xsl:template match="the_geom">
    <xsl:choose>
        <xsl:when test="gml:MultiLineString">
            <centerLineOf>
                <xsl:apply-templates select="gml:MultiLineString/gml:lineStringMember/gml:LineString"/>
            </centerLineOf>
        </xsl:when>
        <xsl:when test="gml:MultiPolygon">
            <extentOf>
                <xsl:apply-templates select="gml:MultiPolygon/gml:polygonMember/gml:Polygon"/>
            </extentOf>
        </xsl:when>
        <xsl:when test="gml:MultiPoint">
            <position>
                <xsl:apply-templates select="gml:MultiPoint/gml:pointMember/gml:Point"/>
            </position>
        </xsl:when>
        <xsl:when test="gml:Point">
            <position>
                <xsl:apply-templates select="gml:Point"/>
            </position>
        </xsl:when>
    </xsl:choose>

```

```
        </position>
      </xsl:when>
    </xsl:choose>
  </xsl:template>

  <xsl:template match="gml:Polygon">
    <Polygon srsName="http://www.opengis.net/gml/srs/epsg.xml#4258">
      <xsl:apply-templates/>
    </Polygon>
  </xsl:template>

  <xsl:template match="gml:outerBoundaryIs">
    <outerBoundaryIs>
      <xsl:apply-templates/>
    </outerBoundaryIs>
  </xsl:template>

  <xsl:template match="gml:innerBoundaryIs">
    <innerBoundaryIs>
      <xsl:apply-templates/>
    </innerBoundaryIs>
  </xsl:template>

  <xsl:template match="gml:LinearRing">
    <LinearRing>
      <xsl:apply-templates/>
    </LinearRing>
  </xsl:template>

  <xsl:template match="gml:LineString">
    <LineString srsName="http://www.opengis.net/gml/srs/epsg.xml#4258">
      <xsl:apply-templates/>
    </LineString>
  </xsl:template>

  <xsl:template match="gml:Point">
    <Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4258">
      <xsl:apply-templates/>
    </Point>
  </xsl:template>

  <xsl:template match="gml:coordinates">
    <coordinates>
      <xsl:value-of select="gisNLS:coordSysTrans(.)"/>
    </coordinates>
  </xsl:template>
</xsl:stylesheet>
```