HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering

Matti Manninen

Using Common XML Schemas for Interoperability in Spatial Systems

Master's Thesis
Espoo, April 18, 2010

Supervisor: Professor Tuomas Aura, Helsinki University of Technology
Instructor: Timo Ruoho M.Sc. (Tech.), Vianova Systems Finland Oy

AALTO UNIVERSITY SCHOOL OF
SCIENCE AND TECHNOLOGY
Faculty of Information and Natural Sciences
Degree Program of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Matti Manninen

Title of thesis:
Using Common XML Schemas for Interoperability in Spatial Systems

Date: April 18, 2010 **Pages:** 10 + 50
Professorship: Data Communications Software **Code:** T-110
Supervisor: Professor Tuomas Aura
Instructor: Timo Ruoho M.Sc. (Tech.)

Abstract

In this thesis I study the process how data can be converted from a vendor specific format to standardized XML format and from there served on a WFS or WMS interface and used by a client program. The XML files used in the study are based on the KuntaGML XML schema.
First I will find out and explain the exact steps that are needed in transferring the data. To be able to modify and publish data, three base points are required: the XML file, the edit program and the WMS/WFS publishing program. The necessity of these will be studied, as well as the steps that are needed in between these stages.
One important issue in the study is how the database should be defined in a situation where the edit program and the XML schema both contain information that is available only to themselves. The goal in this matter is to find an efficient solution where no information is lost when saving the data to the database.
Based on the research, a system for transferring the data is implemented to test the assumptions.

Keywords: WFS, WMS, XML, GML, CAD, Schemas
Language: English

AALTO-YLIOPISTON
TEKNILLINEN KORKEAKOULU
Informaatio- ja luonnontieteiden tiedekunta
Tietotekniikan koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Kirjoittaja: Matti Manninen

Diplomityön aihe:
XML-skeemojen käyttäminen yhteistoimintaan paikkatietojärjestelmissä

Päivämäärä:	18. huhtikuuta 2010	**Sivumäärä:** 10 + 50
Pääaine:	Tietoliikenneohjelmistot	**Koodi:** T-110
Valvoja:	Professori Tuomas Aura	
Ohjaaja:	Diplomi-insinööri Timo Ruoho	

Abstract

Tässä työssä selvitetään kuinka tietoa voidaan muuntaa toimittajakohtaisesta muodosta standardoituun XML-muotoon ja tämän jälkeen jakaa WFS- tai WMS-rajapintojen kautta käyttäjille. Työn XML-tiedostot perustuvat KuntaGML-skeemaan.
Tiedon muokkaamisesta ja luomisesta lähtien tarvitaan kolme asiaa: XML-tiedosto, muokkausohjelma ja WMS/WFS -julkaisuohjelma. Työssä selvitetään näiden vaiheiden tarpeellisuus sekä mitä vaiheita näiden lisäksi tarvitaan.
Tärkeässä osassa on myös tutkimus siitä millainen tietokanta kannattaa määritellä tilanteessa, jossa muokkausohjelma ja XML-skeema sisältävät kukin osittain sellaista tietoa, jota toinen ei sisällä. Tavoitteena on löytää tehokas ratkaisu, jossa tietoa ei häviä talletettaessa sitä tietokantaan.
Tutkimuksen pohjalta työssä implementoidaan tiedonsiirtojärjestelmä, jolla voidaan varmistaa oletuksia.

Avainsanat: WFS, WMS, XML, GML, CAD, Skeemat
Kieli: Englanti

Acknowledgments

I would like to thank everyone who has helped me during the writing of this master's thesis - in particular the following organizations and people:

Vianova Systems Finland Oy and especially Timo Ruoho (Vice President, Technology) for offering me the opportunity to write this thesis, arranging me the needed time to do this and for guiding and commenting the thesis.

Professor Sasu Tarkoma who was the original supervisor of the thesis and who helped with it until the end.

Ville Herva and Jaakko Kuusela for excellent comments and help with proof-reading.

Espoo, April 18, 2010

Matti Manninen

Abbreviations and Acronyms

CAD	Computer Aided Design
GIS	Geographical Information System
GML	Geography Markup Language
KuntaGML	A project aiming for an interoperable data transfer format
Object Data	AutoDesk products specific user-editable Extended Entity Data
WFS	Web Feature Service
WMS	Web Map Service
XDATA	Extended Entity Data
XML	Extensible Markup Language

Contents

List of Figures

Chapter 1

Introduction

In this chapter I will describe the problems that this thesis aims to solve. I will also introduce the domain and the structure of the thesis.

1.1 Purpose of This Thesis

This thesis aims to clarify the things that need to be taken into account when planning and deploying a system with spatial data conversions and a WFS server. The related implementation is about converting spatial data from layer-based[25], CAD-like, database to a standardized XML format and vice versa. From the XML file, also the conversions to WFS and WMS interfaces are described.

1.2 Problem Background

In most software markets there are many vendors that offer solutions for similar purposes. Making these solutions able to understand each other is in no way a trivial task. Without a common data format, every solution would have to be separately made to read data from each other solution. That would be very laborious if there are many vendors. By using a common data format, each of these vendors only needs to implement one two-way conversion to read data from all other vendors. The result from the market point of view is that there are less obstructions for entering the market[1] and thus there is more competition [9]. From a software vendor's point of view

[1]Also means that the users are able to change from a vendor to another more easily.

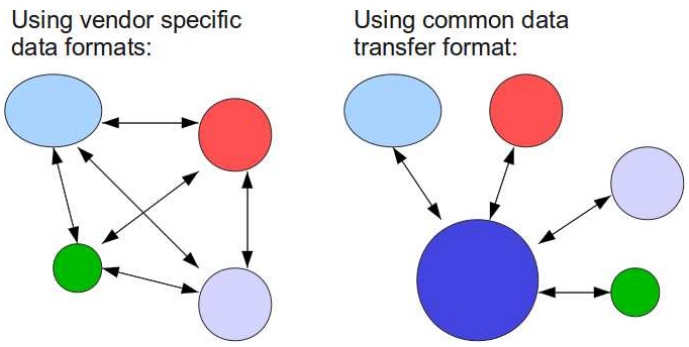

Figure 1.1: Using Common Data Formats Reduces the Total Amount of Conversions

this means lower development and maintenance costs for a solution with good interoperability. Especially for new solutions the common data format might provide a good starting point when designing a database[2].

1.3 Structure

In the chapter 2 Background I explain the techniques and concepts that are important to know when reading this thesis. The chapter 3 Process contains a general description on how to convert XML data and make it available through WFS or WMS interfaces. The chapter 4 Implementation describes the implementation environment, involved software systems and most importantly the techniques that I found suitable for solving the defined problem. The chapter 5 Experimental Results contains the measurements that show the performance one can expect using these techniques. The results are also analyzed to find out the advantages and disadvantages of this implementation. The chapter 6 Conclusion contains a short executive summary about the findings of this thesis, as well as some suggestions for further research.

1.4 Relationship to Related Work

Shanmugasundaram J & Co have studied one part of this thesis' problem from a broader perspective in their article "Efficiently publishing relational data as XML documents"[20]. Their study covers the general case on filter-

[2]Files can be thought to be databases as well

ing data from a relational database and building a matching XML file most efficiently. In my thesis the focus is more specifically on finding an easy way to convert data from a very simple database[3] to a XML file with a complex schema.

Shekhar S & Co studied filtering spatial data by querying GML[4] files in their article "WMS and GML based interoperable web mapping system"[21]. The study is from year 2001. The WFS 1.0.0 specification wasn't released until a year later and therefore the article holds no mention to it. The actual content about a web server that delivers GML output, however, seems very similar to WFS and is thus highly relevant to this thesis. The results of the article could be used for implementing a spatial filter that could for example used to extend the spatial filtering capabilities of the database, if the chosen database doesn't support all of the needed functions. In this thesis all the spatial filtering is done at the database or manually at the CAD software.

[3]CAD drawings are basically databases and in this case can be thought to have very few relations and tables

[4]See Section 2.3

Chapter 2

Background

In this chapter I will introduce the basic concepts and techniques that are relevant in this thesis.

2.1 Spatial Data

Spatial data is "information that identifies the geographic location of features and boundaries on Earth, such as natural or constructed features, oceans, and more. Spatial data is usually stored as coordinates and topology, and is data that can be mapped [24]". The data contains information about the feature, such as type and properties, and position data - points in a specified coordinate system.

2.1.1 Some Uses for Spatial Data

Spatial data is used for charting terrain, planning areas for housing or calculating a route on GPS navigator. In one effort to make the huge amount of existing data more easily available, the European Union has released the Infrastructure for Spatial Information in the European Community (INSPIRE) Directive [4]. The directive dictates that the municipalities in each member country must make the spatial data, that they own, available for purchase by other parties. The idea is that having a infrastructure for exchanging the data would also increase it's usage. The directive should also help to avoid situations where for example two different parties would chart the same area.

In Finland this directive is being implemented according to the JHS 162 - recommendation[7]. The KuntaGML project was started for creating a stan-

dard format for Finnish map and area planning data and for making the most important Finnish software vendors support this format. The KuntaGML project is explained in more detail in Section 4.1.

2.1.2 Coordinate Systems

Coordinate systems make it possible to relate the data to other spatial data. Another term for coordinate system is spatial reference system[22].

You could for example create your own coordinate system to map the interior of your office building. You could decide to call the most south-western corner the origin, (0,0), x-coordinate increases when going east and y coordinate increases when going north. Then you would decide that the distance units are meters.

With this kind of coordinate system, you could map any object in your office with centimeter-scale accuracy using four[1] digits (two of which are decimals) for each coordinate. Using any global coordinate system, you wouldn't probably even find the building if you only used four digits per coordinate[2]. However, with any global coordinate system you could map the object in the building with same accuracy, but you'd most probably need much more digits.

The above example shows the importance of using a proper coordinate system for a proper task in order to reduce the amount of unneeded data. If you already know an object is in a building, whose location is already known, it's useless to spend several more digits per coordinate per object to tell that. Instead, you can just use a coordinate system that offers coordinate transformation equations, just in case you after all need to map the objects globally.

Another advantage of using local coordinate systems is the fact that all two-dimensional expressions for spatial data cause some kind of distortions to the viewed data. Because Earth is a sphere[3] and these coordinate systems try to express the data on a flat surface, like screen or paper, there are bound to be differences on how the data looks at different places. If you are using a global latitude-longitude-based coordinate system and you set your viewer to draw the things properly near equator, the data near the poles will look stretched sideways. On that scale the curvature of the earth will cause the distances to

[1] Assuming the office is smaller than 100m×100m

[2] Assuming the building is at a random location.

[3] Or an ellipsoid to be exact, complicating the matter even further

vary quite a bit. If you use a coordinate system that is meant for a smaller area, the curvature is small and the viewer can show it so that the distortions also remain small. Usually the viewpoint is at a fixed point, depending on the used coordinate system. The other way would be to rotate the viewpoint around the map, so that the "camera" always looks down toward the center of the earth. This doesn't help if there is data both near the equator and near the poles and both are viewed at the same time.

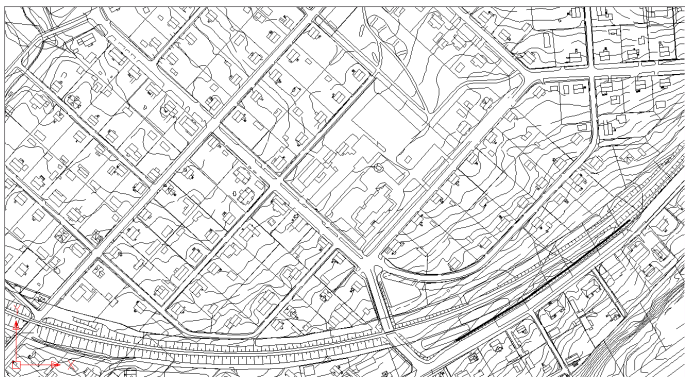

Figure 2.1: Part of city of Lohja in the Finnish Coordinate System KKJ2, Viewed with Autodesk Map 3D

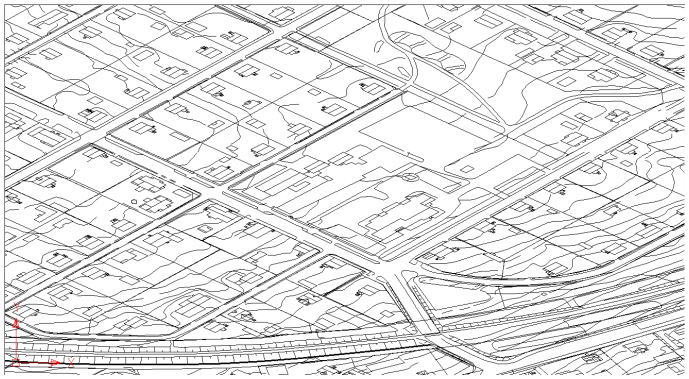

Figure 2.2: The Same Part of Lohja in the Global WGS84 Coordinate System, Viewed with Autodesk Map 3D

Data from any coordinate system with known transformation equations can be converted to other such coordinate systems. However, these conversions often cause some data loss of different kinds. For example, when converting data from a three dimensional coordinate system to a coordinate system with

only two dimensions, the third dimension is lost. Afterwards, the lost data can no longer be returned when converting back to the original coordinate system.

Also, the transformation equations are not always perfect. They may cause some distortions into the data. These distortions can be calculated and they are negated when converting back to the original coordinate system. When converting to some other coordinate system the distortions might not be negated. This might cause the data from the first coordinate system to be displayed differently in the final coordinate system depending on if it was converted directly or through another coordinate system. For example the transformation equations from EUREF-FIN(WGS84) to ETRS-TM35FIN are defined by calculating the distortions at many fixed points on map and minimizing the total distortion using Delaunay triangulation [6] [5]. There are distortions in the end result but they are known are each map position. At the tip of each Delaunay triangle the coordinates are exact with no distortions but all other points have some distortion. Still, all the data can be converted back to the original coordinate system with no data loss.

2.2 Extensible Markup Language (XML)

Extensible Markup Language is a widely used flexible text format, that can be used to present data in tree-like hierarchal structures [26] [8]. In an XML file, the data is stored as text, usually in human-readable format [8].

2.2.1 XML Schemas

XML Schemas "provide a means for defining the structure, content and semantics of XML documents [27]". They can be used to limit the possible elements and their values to only those that the receiving software can use.

2.2.2 Validity and Well Formedness

By definition, an XML file is an XML file only if it is Well Formed [10]. This means that it follows the XML syntax and contains no syntax errors.

An XML file may contain a reference to a schema it claims to conform. If the file contains this reference, it can be checked for validity. The validation is a process of comparing the data in the XML file to the information about

that element in the schema. If for example the data is of wrong type - like a string instead of an integer - then the file isn't valid. In such case, after the check the program that is trying to use the file would know that there is something wrong with the data and it should perhaps abort or at least be cautious.

The schema file can usually be fetched from the Internet from the address that is announced in the referencing XML file.

2.3 Geography Markup Language (GML)

The OpenGIS® Geography Markup Language Encoding Standard (GML) "is an XML grammar for expressing geographical features [14]". It defines a common way to express geometries in XML files.

2.3.1 Using GML

The usual case is that a schema has only a few needed geometry types, for example points, lines and polylines. In such case the user would define the schema as usual, and then in the places where geometries are needed, make a reference to a suitable GML geometry element. It may also be needed to inherit some higher level schema elements from certain GML elements in order to make it possible to read the GML data in a common way. This way a generic software can be used to read the geometries. For example Autodesk AutoCAD MAP 3D can read data from a GML file [1] [19] and import the geometries to descriptive layers, based on the object types in which they belong to in the GML file.

2.3.2 GML Versions

The there are two GML versions that are relevant to this thesis: versions 2.1.2[16][4] and 3.1.1. From this thesis point of view, the biggest difference between them is that the version 3.1.1 allows curves, while 2.1.2 doesn't.

[4]There is a inconsistency with the schema and the specification: In the WFS 1.0.0 schema, the version 2.1.2 is imported, but according to the specification "application schema definitions shall conform to the OpenGIS Geography Markup Language(GML) Implementation Specification, version 2.1.1"[11]

2.4 Web Feature Service (WFS)

"The OpenGIS® Web Feature Service Interface Standard (WFS) defines an interface for specifying requests for retrieving geographic features across the Web using platform-independent calls [15]". A service that implements a WFS interface, can be used for browsing spatial data and the property data attached to it. The user can use this to visualize the data by using a WFS client program that converts the GML geometries, that the server sends. He can also browse the properties of a geometry or save the data for later use. The service can also be used by specially designed clients that can process the data and for example calculate some statistical information from it.

2.4.1 GetCapabilities

The GetCapabilities request is used to find out the available feature types, their bounding boxes and some general data about the service and the available functions, such as available filtering operations and coordinate systems [11],[12].

2.4.2 DescribeFeatureType

The DescribeFeatureType request is used to find out the attributes and other information for a specific feature type. Based on this information, the receiving software knows for example what data types to expect for a certain attribute. The same could be achieved by fetching the schema that is mentioned in the GetFeature response, assuming that the schema can be downloaded. The advantage in using DescribeFeatureType instead is that only the needed parts of the schema can be delivered. In addition, the service provider can mirror the schema to their own server to ensure that the schema will be available[5][11],[12].

2.4.3 GetFeature

The GetFeature request is used to ask for the actual spatial data. The request contains the feature type names that user needs. In addition to that, the user

[5]For example in cases where the schema is from a third party and is hosted at a different location.

can also use a bounding box or properties to filter the results, if the server supports these operations [11],[12].

2.4.4 Versions

There are two commonly used versions of WFS: versions 1.0.0 and 1.1.0. From this thesis point of view the biggest difference is that version 1.0.0 uses the GML version 2.1.2 and 1.1.0 uses the GML version 3.1.1. This means that WFS version 1.1.0 supports curves and 1.0.0 does not.

2.4.5 Filtering

Filtering is the process of selecting only the proper objects from the database. The parameters are received from the user in the query. Some of these parameters are mandatory, like the Feature Type, and some are optional, like the Bounding Box and any properties that should match the given value [12],[11].

2.5 Web Map Service (WMS)

"The OpenGIS® Web Map Service Interface Standard (WMS) provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases [17]". In practice, the user can use a WMS client to fetch raster images of the data. The server sends an image of the certain area, that the client program requests. The client may also request for the properties of the whole service. Individual objects are not sent as geometries nor do they have any properties. Only the image of the data is transferred. This makes implementation of the WMS clients a lot easier than the WFS clients, as there's no need to first convert the GML geometries to any other data form, and after that drawing them. In WMS the image can just be shown as is. The service provider can also select the drawing styles that they consider best suited for the data. On the downside, there is less information available for the client, because there aren't geometries with precise points and properties.

WMS is quite popular way to show map information. Probably the biggest advantage of WMS over WFS is that it doesn't need to transfer as much data when looking at a wide area of data. In other words, the size of the raster

image doesn't depend on the zoom level. The different zoom levels can be cached on the server, which also helps to keep the server load unrelated to the amount of data in the database. One other advantage is that the raster images don't have too much precise and easily reusable information, which helps in keeping the valuable complete data hidden.

2.5.1 GetCapabilities

The GetCapabilities request is used to find out the available layers and their styles, as well as some general data about the service. The layers in WMS are similar to the feature types in WFS. With layers there is more freedom to combine different real world object types to a single layer, for example the railroads and also the related traffic control signs. The server may offer several styles for each layer [13],[3].

2.5.2 GetMap

The GetMap request is used to fetch the actual raster image. The mandatory parameters for GetMap request control for example the following[6] [13],[3]:

- Layers and their styles
- Bounding Box
- Image width and height
- Image format

2.6 Computer Aided Design (CAD) Software

CAD software is used to handle point, line, curve, polygon, etc. data[7] in three dimensions. This kind of point-based data can also represent spatial data.

[6]For full list of possible parameters, see page 33 in [13]

[7]All these geometries in the drawing are called Entities. Entity (usually) contains a geometry but also contains other information, like the layer and a unique handle

2.6.1 Classifying Different Objects

Typically a CAD program has layers that the user can use to distinguish the different types of objects from each other. For example he could have one layer that contains trees and another one that contains road center lines. The layers can be turned on and off individually and their drawing details, like color and line type, can be defined.

Some CAD programs have ways to define an Object Class for an object. This is a more specific method than just using the layers. Setting an Object Class may be used to add the proper attributes for that object. For example in Autodesk Civil 3D, adding an object with the predefined Object Class "Pipe" will add an object with a data attribute "Inner Diameter" to a specific layer. One could say that objects with an Object Class are strongly typed objects. They can be forced to contain certain attributes that have specific allowed values.

2.6.2 Extended Entity Data (XData)

Some CAD programs have ways to add generic data to an entity. In Autodesk AutoCAD and Map 3D programs one of these ways is called Extended Entity Data (XData). The XData is unstructured key-value type data. Additionally, Application Name property must be set for all XData entries. The XData is mainly invisible and undisturbing for the user, though there are commands he can use to see the data. A programmer can use XData to add string- or numeric data to an entity. The XData is saved in the drawing so it can be used to store information from one session to another.

The XData could be used for example to distinguish the objects that have been added on a certain date. Many persons may be using a program to add entities to a single drawing. If each program adds the date to the XData of each entity they add, then everyone can read the date of every entity in the drawing.

2.6.3 Object Data

Autodesk CAD products, including AutoCAD and Map 3D, also have possibility to add Object Data to objects. This is mostly similar to XData, being unstructured and object related key-value type generic data. The exception is that the user can see and edit the Object Data as any other of the object's properties.

2.6.4 Coordinate Systems in CAD Software

Some CAD products are meant for use with spatial data and understand different coordinate systems. Autodesk Map 3D is one such product. A drawing can be assigned to a certain coordinate system, which defines how the data is viewed and sets the valid coordinate ranges for the drawing[8]. When importing data to the drawing the program will automatically convert data to the assigned coordinate system if the source coordinate system is known. Also if the assigned coordinate system is changed, the program automatically converts such data to the new coordinate system.

As example, the Figures 2.1 and 2.2 were done using Autodesk Map 3D. The new drawing was first assigned to KKJ2 coordinate system. Then the data was connected from a Oracle database which also uses KKJ2. The data was seen as in the Figure 2.1. The coordinates were around (2 500 000, 6 700 000). The assigned coordinate system was then changed to WGS84. After this the outlook of the data changed to that of the Figure 2.2 and the coordinates changed to around (24, 60).

[8]For example, you cannot validly enter y = latitude 91 °N in Lat-Long coordinate systems.

Chapter 3

Process

This chapter explains the steps that are required in publishing and using the data through WFS and WMS interfaces. The Figure 3.1 visualises the process of converting GIS data to the XML, WFS and WMS formats.

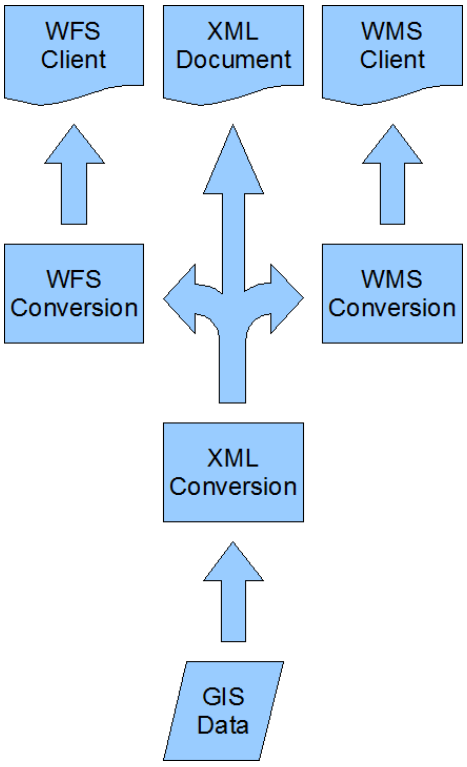

Figure 3.1: Converting GIS Data to the XML, WFS and WMS Formats.

3.1 The XML Conversion

The XML Conversion is the process where data is translated from a vendor specific software to the common XML file or vice-versa. I will focus on conversions in a layer-centered CAD-type environment.

3.1.1 Basics

In a CAD drawing, the properties that can be translated to the XML file, are in the layer, geometry type and XData of each geometry. The data hierarchy in the XML file is in a sense backwards compared to a CAD drawing. The structure contains a list of a certain type of objects, that usually have a geometry. The schema doesn't necessarily require that all the objects contain a geometry though. For example labels in XML, which correspond to text type objects in a CAD drawing, are placed under a different element than other actual (or GML) geometries. Therefore if there is a text type object in a CAD drawing, and there is no way to link it to some other object, it will have to be put in the XML file so that the element only contains the label insertion point, and not an actual geometry.

The above also pictures the problem that the unhierarchal nature of a CAD drawing presents. Linking related CAD geometries together from a picture that doesn't contain previous connections is very difficult. This would be a needed if one wished to correctly insert them as one element into an XML file.

3.1.2 Mapping

Mapping is the process of translating XML paths, element names and their values to different properties in the target software. In this thesis, I will call this kind of mapping, from XML to vendor specific software, Forward Mapping, and the other way around, mapping from vendor specific software to XML is referred to as Reverse Mapping.

In Forward Mapping there must be a separate mapping for each path and all combinations of its elements' values, that can lead to a different outcome in the target system. For example, a building with usage type "housing" might go to layer "house", but a building with usage type "housing" and height "20 meters" would go to layer "apartment building". Also, a building with usage

type "housing" and height "5 meters" might also be mapped to the same layer "house" that the building without height information was mapped to. This gives a good hint of the challenges that will be faced when trying to Reverse Map the same data back.

Reverse Mapping is the same kind of process, just the other way around. The geometry related data like layer, block reference values and perhaps the XData properties, is mapped in order to find out the destination path in the XML and often some sub-elements and their values.

3.1.3 Challenges of Mapping

In the example in the previous section, it would be impossible to Reverse Map objects from layer "house" back with their original properties. This shows the typical case of data loss in the conversion: the data has to be mapped to a more general form, after which it can no longer be returned to the original specific form.

In the case of Forward Mapping from XML to CAD drawing, there would be one way to avoid this kind of data loss. The properties that would be lost in conversion could be serialized and added as XData to the CAD drawing objects. However, this approach has its risks because of the typical ways that CAD based software is used. Quite often CAD objects are copied and edited to get a new similar object. When an object is copied, its XData will also be copied to the new object. This would likely lead to false information when Reverse Mapping because the user typically won't see the XData of either object. Thus he won't know that the house he copied happened to be assigned with the Height=10m property while he actually wanted to add an 5m house but just had no way to specify it.

The advantages of using XData are limited to begin able to Reverse Map the data in the same form as it was before Forward Mapping. Without changes to the CAD based software, it won't be able to use the additional data.

Object Data could be used instead of XData to store the properties. It is visible to the user and is also user-editable. The disadvantage is that showing it to the user adds to the complexity of the program. The program would appear different depending on the source of the data. The data from a vendor that writes the heights to the XML file would show the heights on the

objects while the rest of the similar objects would not. From programming point of view, it would be quite difficult to determine what properties to add and what not to add. Some properties might be such that they would usually be generated during the Reverse Mapping. Some might be text justification or such data that is already on the geometry. Also there would have to be a way for the user to see what properties he can add to an object that doesn't previously have those. This would require runtime awareness of the mapping of the selected object and awareness of the XML schema to know the available properties for that element.

Using Object Data too makes it possible to Reverse Map the data in the same form as it was before Forward Mapping. In addition the user is able to add new information that can be written to the XML file when Reverse Mapping. Using Object Data would require many changes for the software to still be usable. Using Object Data changes the CAD based software's data format toward the XML data format, which is good for interoperability. Unfortunately it does this in an unstructured manner which could cause problems when the XML schema changes. If changes are made to the CAD based software, it might be better to change the data format in a structured manned, so that the data could be mapped without data loss, instead of using unmapped data trough Object Data. Even with its disadvantages, Object Data is a very potential way to reduce data loss improve interoperability.

Data loss at Reverse Mapping time could be even more challenging to avoid. A standardized XML schema as common data format is likely more difficult to change to suit the needs of one vendor. This can easily result to a schema that simply has no place to put some data that is available in the CAD drawing. The XML file has strict requirements about the allowed and needed elements and properties; if those are not met, the file doesn't conform to the schema and is not valid. However, if an element has a free text field of some kind, that can be used to serialize the additional data to the element. Also any elements with unspecified real-world meaning and a free text field can be used to write unsupported objects. If the actual element doesn't have a free text field but does have some kind of id field, the id can be used to tie an unspecified element to it. In this case the additional data can be put to the unspecified element.

All the above "tricks" to write data that isn't in the schema, have the problem that the solution will be vendor specific. There won't be a standard way for other vendors to read and understand the additional information. Giving too much liberties for these methods in schema definition might risky for

the schema's main goal: interoperability. If writing unspecified data is much easier, some vendors might take the easy route and not try to find a way to change their own software to match the schema.

In any case the XML file is usually needed when transferring data from one vendor to another. Therefore it might not be worth the risks to avoid data loss in conversions from one vendor software to XML and back to the same vendor software. Especially from the schema maker's point of view, this is something to consider.

3.1.4 Conversion of Hierarchical Data

In order to be able to map the objects from CAD drawing to an XML hierarchy, the relations must be somehow expressed in the CAD objects. In this study I used XData for adding unique IDs and parent IDs to the objects. If and object is on third level on hierarchy, then also the grand-parents ID etc. could be added for best performance. With the information about the ID and parent ID, several CAD objects can be added under one XML element.

When generating the XML file there are two choices: go through the objects hierarchically one level at the time or go through them in random order and add each object to its proper level in the hierarchy.

Adding Hierarchically Sorted Objects

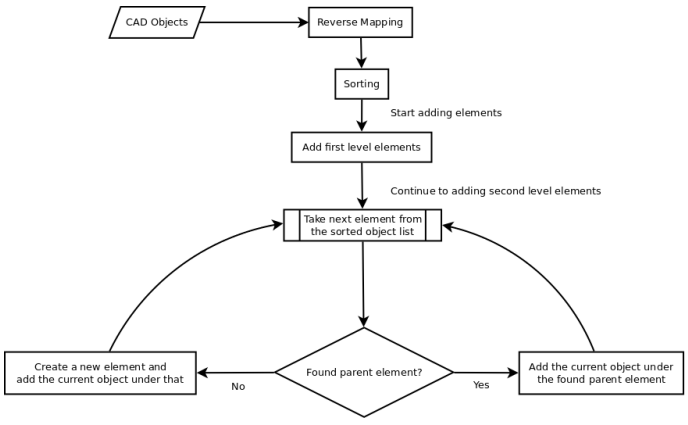

Figure 3.2: Adding Hierarchically Sorted Objects to XML Document.

Going through the objects hierarchically would require that the objects were first sorted accordingly. In the CAD drawing the objects are in a seemingly random order and the level in hierarchy has to be resolved by Reverse Mapping. After mapping, the objects can be sorted according to the count of the / characters in the mapping result path. By adding one level of hierarchy at a time, one knows that the converted parent object should always be found and the current object can be added under it. A third level object that has both parent and grand-parent IDs can be added by first searching the first hierarchy level and then only the child elements of the matching first level node. In this method only the ID and parent ID are needed for begin able to map any hierarchy. Grand-parent ID[1] just makes finding the correct parent XML element easier. Without grand-parent ID one may have to search through all the objects lower in the hierarchy.

Adding Objects in Random Order

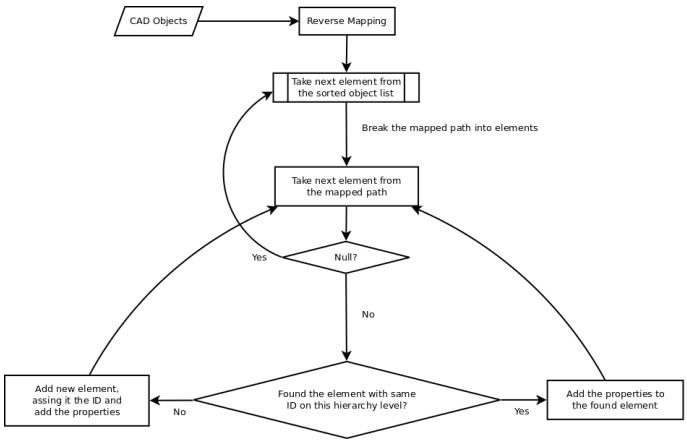

Figure 3.3: Adding Randomly Ordered Hierarchical Objects to XML Document.

Adding the objects in random order has the advantage of only having to go through the objects once. The disadvantage is that there has to be a parent ID for each higher hierarchy levels to find the object the correct parent. Again the process starts with Reverse Mapping. The resulting path tells all the elements that need to be either added or found to add the last level element or its properties. One level at a time, the program tries to find an element with ID that matches the ID for that level from XData. If none is

[1]Or any parent ID even higher in the hierarchy

found, then a new one is added with that ID. This way the last element will then have all the parents with proper IDs. Later when the program encounters an object that is mapped to a higher level in the hierarchy, the element with matching (actual) ID will be found. In that case, the properties are added to that element without adding a new element. In layman's terms the program fills in the blanks.

In both styles the object that have no parent ID are added by creating new elements to all previous levels. If the object has ID for itself, its added in any case.

3.2 From XML To WFS

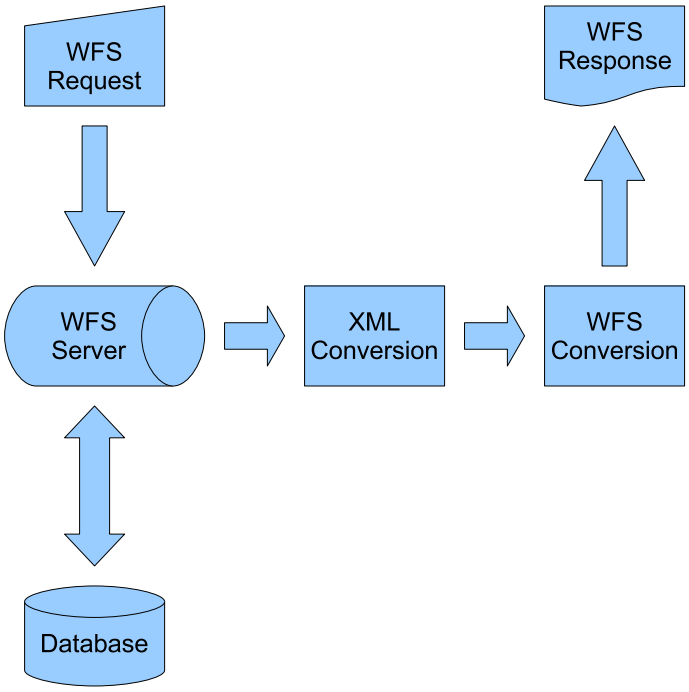

Figure 3.4: The WFS Conversion Process.

In general the conversion from XML to WFS is quite straight forward. The WFS response for GetFeature[2] contains the main data that was in the XML. The data for WFS format can be selected so that you'll just have

[2]Explained in Section 2.4.3

to filter all nodes from certain level of the XML hierarcy and add each of them in to the WFS frame. Each of them are added to root element wfs:FeatureCollection under a gml:featureMember object. Therefore an object /my:root/my:objects/my:object in XML could translate to /wfs:FeatureCollection/gml:featureMember/my:object in the WFS response.

Of course there are some requisites for a proper conversion. Mainly the WFS must contain references to schema locations which contain definitions for the objects. Only the objects which can be contained in the WFS need to be defined, meaning that in the former example only my:object and all its subobjects and types need to be defined, where as my:objects wouldn't need to be defined. This means that one can save bandwidth by using only the needed feature types. In practice though, not all clients need the schemas.

3.3 The WMS Conversion

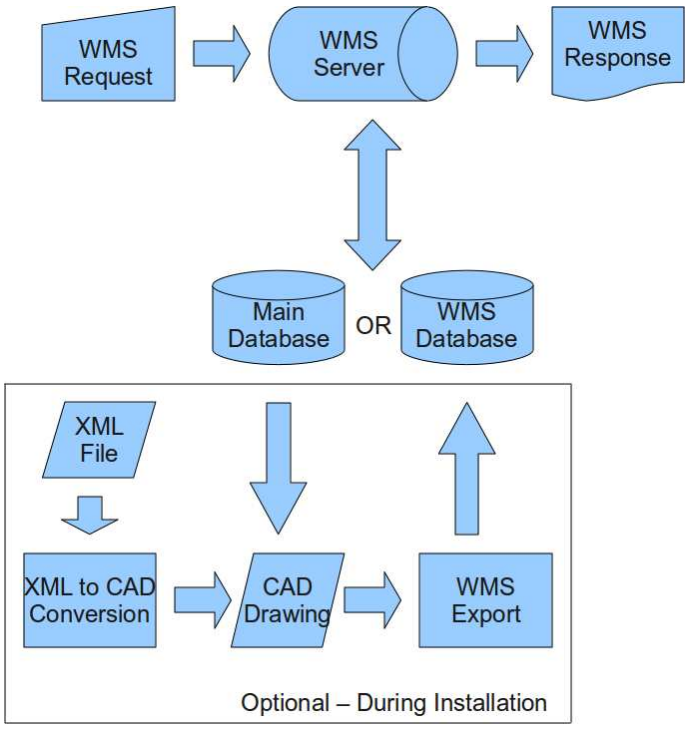

Figure 3.5: The WMS Conversion Process.

The WMS conversion from a CAD drawing is very simple. In the simplest case the CAD program is ordered to take pictures of the data on different

zoom levels. The WMS server program then reads these pictures, crops, combines and perhaps zooms them so that it can deliver the requested area to the user.

Conversion straight from the XML file on the other hand is more difficult. The XML files in question don't have definitions for the outlook - they only contain the data, not its presentation form. The most sensible real world approach would probably be to read the files into a CAD program, set the presentation form there and then export to WMS with the CAD program. Also, WMS servers, such as Autodesk Mapguide can themselves read CAD drawings and use them to generate the images on the fly. Using database geometries is also possible, but then one must define the presentation form.

Chapter 4

Implementation

This section focuses on the properties of the actual system that I implemented during this project. I will describe and analyze the phases that are done in order to get the data from XML to the WMS and WFS publishing. The Figure 4.1 visualizes the process structure of the basic implementation that does the necessary conversions.

4.1 The KuntaGML Project

The main goal of the KuntaGML project is to make area planning and map programs from different manufacturers interoperable. Currently there are several different systems in use in different local governments of Finland and in consulting companies. Transferring data from system to another is difficult and the conversion usually requires manual labor. For example fusions of local governments might create a need for transferring data from system to another in large scales.

The KuntaGML project aims to enhance interoperability by introducing a common XML data schema that all the participating vendors will support. The project also defines that the data must be publishable using WMS and WFS interfaces in certain format.

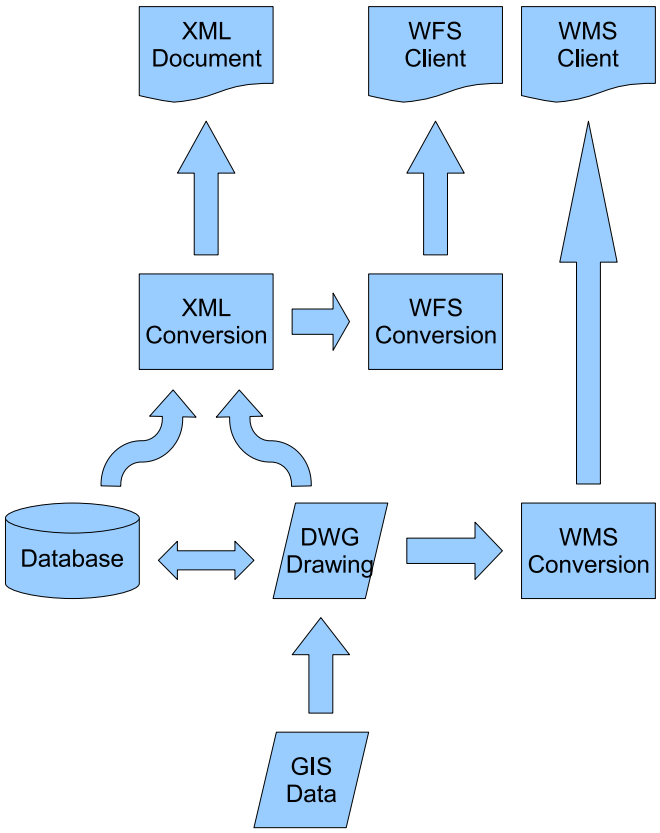

Figure 4.1: The Conversion Process in Implementation with Basic Database

4.2 The Involved Software Systems

All the systems involved in the KuntaGML project have much in common. The most unifying factor is that all of them more or less use layers to tell the object types apart. There are differences in the degrees in which the property data of the objects is contained either in layer or in object data or such. Also some programs have hierarchal dependencies between the objects while others focus on on-the-fly processing that is based on the locations, layers and texts of the objects.

In the case of KuntaGML there are many different vendors that have similar data but it is stored in a different format and handled differently. These differences create many challenges in designing the schema, as well as in implementing the best compromise. Also, it is likely that the vendors each have, in one place, something more than the common schema defines and in another place there isn't enough data to match the schema. This could cause problems if some data needs to be generated in order to match the schema and the another program actually uses that data and presumes that the data is real.

4.2.1 Testing tools

These products and standards that were used in the implementation:

- KuntaGML Asemakaava[1] and KuntaGML Kantakartta[2] schemas and XML data files based on these schemas. The GML standard is used in these schemas.
- Autodesk AutoCAD MAP 3D, Novapoint Area Planning and Novapoint MAP for editing the data.
- Oracle 10g Standard Edition with Locator functionality as a database for the edited data, as well as for the publishing program.
- Autodesk MapGuide Enterprise 2009 for publishing the data. MapGuide offers WMS services.
- The Carbon Project Gaia for viewing the WMS and WFS data.

[1] Translates to MunicipalityGML Area Plan
[2] Translates to MunicipalityGML Map

- Mozilla Firefox to make GET method requests and save the responses.
- The Wireshark Network Protocol Analyzer to view and save the requests and the responses in both GET and POST method requests.
- Altova XMLSpy to verify the requests and the responses.
- Microsoft/Sysinternals DebugView for time measurements.

4.2.2 Novapoint Map

Novapoint Map is a program for editing and storing general map data. Such data may include for example roads, buildings, height lines, swamp areas. It is used on top of Autodesk Map 3D and uses mostly the editing functionalities of AutoCAD. The amount of data is usually large (tens of thousands objects in one drawing, millions in total), even though it is usually spread to several files for CAD software performance reasons.

This is a geometry based software. All the information is stored in the geometry and its related properties, such as layer, text, or block name and its properties. The layer is the most important property as it's used to define the type for an object. Text and block name can specify the definition further.

Normally the program doesn't use XData or ObjectData to store information. In case of KuntaGML, some additional properties may be added to XData or Object Data as descriptive power of the current layers is not enough in all cases.

4.2.3 Novapoint Area Planning

Novapoint Area Planning is a program for editing and storing area planning data. Such data is usually quite limited in size (thousands of objects) but contains relations between the objects. The data typically consists of one plan for certain part of a city. For this area the new plan takes precedence. For a full picture of a city's current plans, all the plans must be combined so, that for each point on the map only the newest plan containing it is active.

During the research, WFS and KuntaGML data transfer functionality was developed also for Novapoint Area Planning. However, all the measurements are done with the Novapoint Map data because of its much larger

data amount in the typical use case. Also, the software needs to perform many runtime calculations which are very time consuming compared to the functionality that this thesis focuses on.

Even without the measurements, Novapoint Area Planning is of some interest for this thesis. It contains the relations between objects and the area planning schema (KuntaGML Asemakaava) is hierarchal. Therefore there was a need for using the XData to retain the relations and hierarchies.

The Stored Data

Like the Novapoint Map, Area Planning also holds most of the data in the geometry and it's related properties. The difference is that a lot of the data is calculated in runtime. For example a usage area is defined by placing a text object inside the desired area which is limited by connected generic lines. This means that the lines themselves cannot be set to usage area lines - that would break whatever is on the other side of the line.

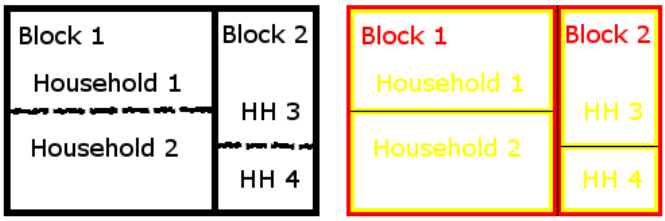

Figure 4.2: An Example of Usage Areas

The Figure 4.2 illustrates the problem. On the left the areas are defined by separate lines of two types. The outer lines define both the block area and household area. On the right the areas are defined by closed lines of specific type and there has to be separate line for each area.

The KuntaGML format needs areas of specific types so the program has to generate them from the generic lines. This results in overlapping areas and lines - something which Novapoint Area Planning as a design software deliberately avoids but KuntaGML as a transfer format needs. At design-time the generic lines are useful because then you only have to move one cross-section point and all the surrounding areas will change their shape and they cannot incorrectly overlap (a piece of land can only be assigned to one household at a time). In a transfer format or database specific areas are

useful because they can be easily filtered. There is overlapping in the data (the piece of land can be assigned to one household and the block that the household belongs to) but if it's not modified in that format, there is no risk of incorrect overlapping. The storage costs of the duplicate data points are a reasonable price for not having to calculate all the areas at runtime.

Additional Data for The Conversion

In addition to the usual mapping that is needed in the conversion process, the program also needs to tie the areas together in a hierarchal fashion. The program needs to add the proper households to the blocks they belong. In this study this was achieved by adding some XData to the objects when the areas are generated. The added XData contains an ID unique for each object. If the object belongs to another object in the hierarchy, then also the ID of the parent and the possible grand-parent is added.

4.3 Database for WFS

The WFS server is expected to be able to deliver detailed data from a selected area while there is a large area from where the selection can be made. Therefore it is important that the database is fast enough in the needed selection queries. In this section I compare a quick basic implementation and a more improved implementation.

4.3.1 The Basic Database

For this project I created a database that allows data to be transferred between the DWG drawing and Oracle database without loss of data in the typical use case. Only the geometries and their related attributes are transferred, layer definitions are not. This means that the data must be read back to a CAD drawing with the same layer definitions if the user wants to retain the outlook.

With a transfer method without relevant data loss, it is easy to use the same operations for converting the data from both formats to XML format.

The database design is geometry focused, as is case with the CAD drawing, that in fact is a geometry focused database too. In addition to the

geometry the database has fields for XData, text and block data. The additional data, for example the text, its size and justification and so on, can be serialized into these fields. The serialization uses key-value pairs with three types of different separators. I tried to select such separators that they would be unlikely to occur in the seriazable data but which are allowed in oracle text fields. Preferably, the separators would be escaped in the serialization.

4.3.2 Filtering with the Basic Database

Using a geometry and layer oriented database presents a problem when filtering the data based on the schema's elements and values. In order to filter data based on the schema (filtering by selecting certain element names or values), the filtering parameters must first be translated to layers. This would work fine if there were only simple - one-to-one - mappings, where one layer[3] corresponds to one element in the schema. Even then, filtering based on the values of the elements could be difficult, as these properties may depend on the type of geometry that the element has. They may also depend on the values of block data or text data, which in this implementation are serialized strings and therefore might not be easy to query by the database. All in all, with a database that doesn't take the schema into account, completely accurate filtering is difficult.

The filtering begins by receiving the parameters, by which the elements, or in this case geometries in database, should be selected. These might be, in KuntaGML schema, the following:

1. Elements of type kanta:Rakennus(Building)
2. Elements that have kanta:rakennuksenKayttotarkoitus Equals "asuinrakennus"(Usage Type is Housing Building)
3. Geometries that are inside this Bounding Box

In order to filter with them, these parameters must first be mapped to the layers. In this case we are talking about Forward Mapping from element names to layers. The two first of these parameters can be mapped to layers. The first parameter could result in layer "Building". The second parameter could result in the layer "House". In addition to these Forward Mapped layers, it could be that also layers "Apartment" and "Cottage" would be translated to the Building-element in the XML file in Reverse Mapping.

[3]or objects of certain type in one layer

The third parameter is a spatial selection query, that is performed based only on the geometries. It too limits the rows that are selected with the filtering query. The implementation uses Oracle Locator's spatial operators, like SDO_RELATE[18].

4.3.3 The Improved Database

In addition to providing the basic capabilities - filtering and showing data - I wanted to improve the performance and accuracy of filtering and to add some data management capabilities. To achieve this, I added three columns to the database: Mapping, Properties and Batch Name.

Filtering Improvements

The Mapping column contains the mapping from a layer to a schema path. This makes it possible to query the Feature Type names directly, instead of selecting a list of layers that sometimes but not always match to the requested Feature Type. This is expected to, by itself, provide some small performance increase, as the amount of geometries that are fetched but not used, is decreased a bit. The bigger reason for adding this column is that it is a prerequisite for the property based filtering. The data may look like this:

> kanta:KantakarttaAineisto/kanta:rakennetut_tilat/kanta:Rakennus/
> kanta:rakennuksenosat/kanta:RakennuksenOsa

The Feature Type isn't always at the same depth as the WFS would hope it to be. For example the above element would be a few levels deeper, as the WFS frame takes the second level elements, kanta:Rakennus in that case. However, this isn't that big of a problem if the SQL query uses wild cards to only take the beginning of the wished path. In this case, if you were searching kanta:Rakennus from the database, you could just search for all entries where Mapping begins with .../kanta:Rakennus.

The Properties column contains the properties that the layer mapper would add to the XML element. These properties are in serialized form, similar to the serialized XData, text and block data. The Properties data for the object in the previous example could look like this:

kanta:tyyppi¤seinälinja#
kanta:parent.parent.rakennuksenKayttotarkoitus¤asuinrakennus#

These are the same properties that would be added to the actual XML elements. The data in this example has the "parent" tag which marks that the property rakennuksenKayttotarkoitus would actually be inserted to the element two levels higher in the hierarcy, to the element kanta:Rakennus. Again, using wild cards in the SQL query, these can easily be selected regardless. The above object could be found with query where Property contains *rakennuksenKayttotarkoitus¤asuinrakennus#*. At the end of the string there must be a separator too. When it's there, the queries can be made so that values with only the same beginning won't match. For example "valueThatShouldntMatch" would match to "value*" query, but not to "value#*".

The above query could cause a problem if there were for example two different kanta:tyyppi elements on two different depths. At such case, the elements could get mixed up, especially with negative searches. This doesn't cause problems with the current schema. Also, the searches could be refined, by using the "parent." tag in the searches as well, so that there wouldn't be problems even in the aforementioned case. In other words, this database structure doesn't limit the usage even in such cases. Besides, at least the Gaia WFS client doesn't make difference between the elements at different depths, which can be seen from the filter in A.2.1.

Searches that use both Mapping and Properties column can provide accurate searches and return only the requested geometries. This is expected to improve the performance, especially in searches using Property Filtering. In such searches the performance should improve the most when there are many objects that have the same mapping but only a few that have the searched property. Previously in these cases, all the objects with that mapping (and perhaps even some objects with some other mapping), would have been fetched, converted to internal format, then to XML format and only then the few with the searched property would have been selected and added to the WFS frame. Now the database will do some more work, but only return those objects that are needed. Converting and adding these few objects should be considerably faster, all in all.

Data Management Improvements

There was also a need for simple data management features, such as partial, batch based updates to the data. For this purpose, I added a column that

contains the batch name for the transfer. The batch name could be for example the name of the DWG file that the data was transferred from. This way when there are some updates in the DWG drawing, the data transfer program would first delete from the database the rows that have the specific batch name assigned to them. After this, it would just insert the new data to the database with the same batch name, effectively replacing the old data.

The advantages of this technique are that it is easy to implement and it offers very few sources for errors in the resulting data. As everything old from the same batch is deleted before the transfer, no old deleted objects can mistakenly exist in the database after an update. Thus the likely errors would be missing data, which could be caused by interrupted connection or such. These problems can be solved quite well by using transactions. The transaction is an atomic operation, in this case for the whole set of changes, containing the deletes and inserts. All these commands go to a cache in the database. Only when the transaction is finally commited, which is the last command, the database will make all the changes using its local cache. If the connection is interrupted at some point, the transaction never gets commited, and therefore no changes are made in the actual data [2]. Also, the data can be checked by querying for the row count after the transfer, regardless of whether or not a transaction was used.

Another advantage in using batch based updates is that there isn't a need for unique identifiers in the image. Such reliable identifiers don't currently exists in the DWG drawings that are used as the data source. The identifiers would also require that add, delete, copy and all other edit functions would be done with non-AutoCAD commands and that they were tracked for easy change-set style transfer.

The obvious disadvantage is that lots of data will be transferred unnecessarily if there are only a few changes.

4.4 Use Cases

Even though this article is mainly focused on the publishing aspect of the data, I'd like to also explain some cases on working with data based on a common schema.

The simplest case for using a common schema is transferring data from one vendor's software to another similar software by another vendor. In such case

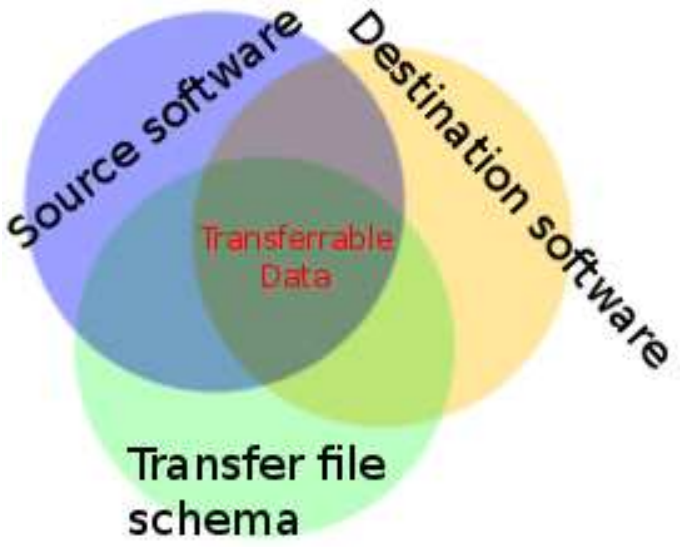

Figure 4.3: Transferrable Data is Limited by Each Involved Software

the first user would just do an export from the original data, send it to the other user by, for example, e-mail. The other user would then do an import and start using the data from his own software. The resulting data in the latter system would then be, at best, the intersection of the capabilities of these two softwares and the schema. This is illustrated in the Figure 4.3.

A more complex use case for the WFS interfaces would be a situation where a data distributer would like to combine data from several geographical locations. To make this possible, each municipality, city or other data source would first have to make their data available on their own WFS services. This could include exporting the data from its main storage to XML files. It could as well be moving the data to a publishing database and converting the needed parts on-the-fly to XML format when requested from the WFS server.

When the data is available on the WFS server, it can be retrieved using a WFS client, a standard one or a custom made. With Gaia WFS client for example, the user can add several layers to the picture, each from a different server, if he wants. This way it's easy to combine the data from different data sources. In addition, the user can perform filtering operations for the data. For example if the user was interested in all the buildings that have more than three floors, he could do so. Such specific filtering could cause him problems though: data from each server is intersection of the capabili-

ties of the original software and the schema. Because of that, the data that is available for each part, would differ. As a result, he would get only the 'more than three-story' buildings that have their floor count defined in their original software.

Chapter 5

Experimental Results

In order to find out the actual speed differences in the various involved process steps, I measured the performance in some typical use cases.

5.1 Arrangements

The GetCapabilities command was executed from Firefox on remote computer by entering the URL <`http://host/service.aspx?service=WFS&version=1.0.0&request=GetCapabilities`>. The times were measured from the server using System.Diagnostics.Debug.Print commands and a debug build of the WFS server implementation. DebugView program was used to catch these debug messages. The server wasn't in completely dedicated use, but the load on the server was relatively low. The server was a VMware virtual machine with Windows Server 2003 Web Edition. The server and the remote computer were connected to each other with 1Gbit ethernet. The WFS server and the Oracle server were also connected to each other with 1Gbit ethernet.

Each test was executed three times so that abnormal results could be detected, analyzed and left out if needed. These abnormal results can occur when the server performs some other tasks than the measured one. The effect of these errors is much larger when the time to be measured is short.

In some cases there can also be seen some considerable variance in the performance, that is most probably caused by the database using a cache. This can be seen in a case where the geometries are fetched for the first time after

inserting them. The first search is considerably slower than the following searches. During the tests, I encountered a difference of 2.5 to 4 times slower than the subsequent searches.

5.2 The Basic Database

First I performed the performance measurements using the Basic Database to see which tasks I should focus on improving.

5.2.1 GetCapabilities

Upon receiving a GetCapabilities request, the WFS server checks the available feature types from the database and fetches a Bounding Box for each feature type. According to my measurements, there are two time consuming steps in the operation.

Fetching Feature Types

The first database operation is fetching the available feature types. The time that is used for this operation is very small, 37ms to 60ms with the test data. The time consumption per geometry gets lower with more geometries. Amount of different layers and layer mappings will most probably increase the time consumption more. In any case, the time consumption is about one thousandth of the time consumption of the slowest operation: fetching the Bounding Boxes. In the tests there were some results that clearly appeared to have suffered a delay and these results were left out from the averages. These can be seen red on the Test Record A.1.1 on the Get Layers operation. The Figure 5.1 shows the measurements as graph.

Fetching Bounding Boxes

According to the measurements, the Bounding Box performance seems to be very heavily dependent on the amount of total geometries. If the connection between the database and server is slow in terms of latency, the performance would be also dependent on the amount of available feature types, as every feature type would require its own query (with the tested implementation). In these measurements the communication delays seem negligible compared to the other delays in that process. The time usage is linearly dependent on

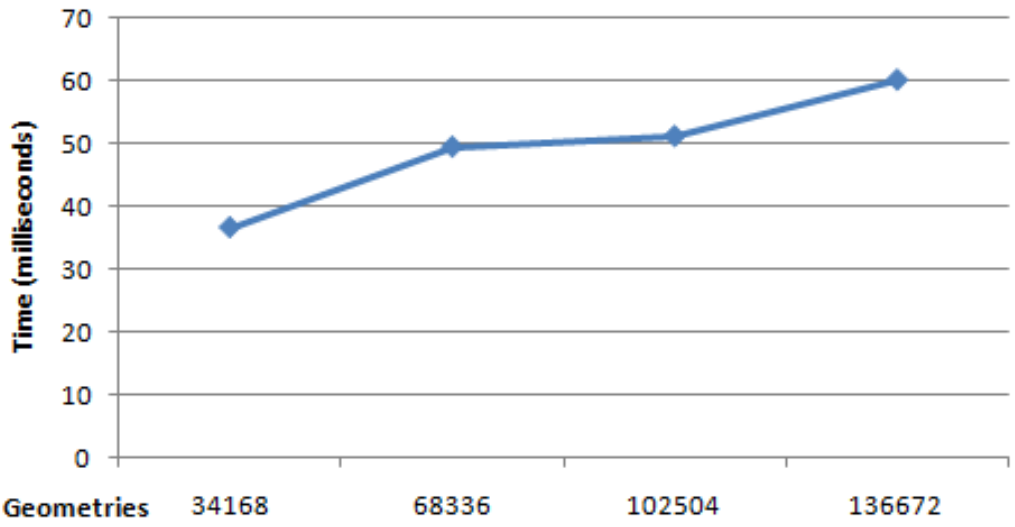

Figure 5.1: Basic Database: The Performance of Get Feature Types Query

the number of geometries and the performance would seem to be of O(N) efficiency or better. The absolute times, around 78s for 137 000 geometries, ensure that this kind of implementation couldn't be used in a real environment. The Figure 5.2 shows the measurements as graph.

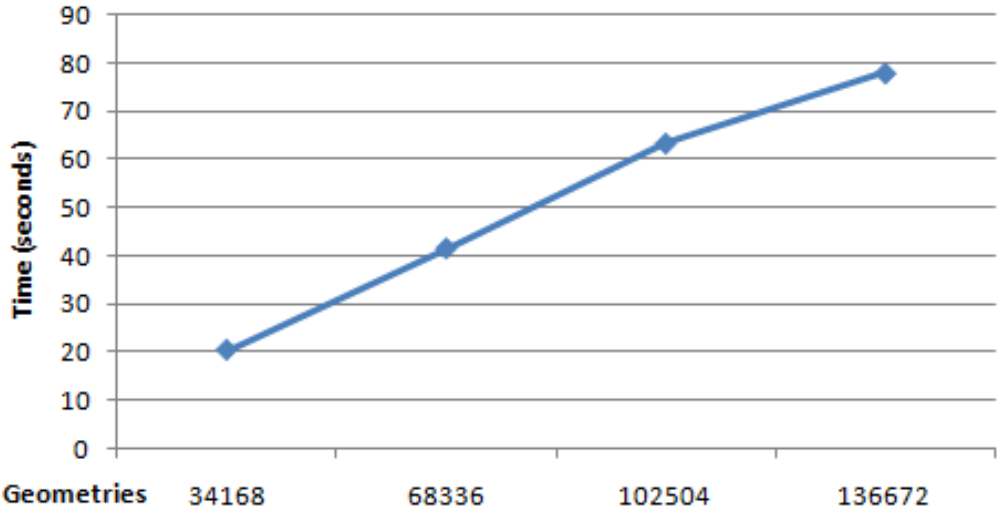

Figure 5.2: Basic Database: The Performance of Bounding Box Queries for Feature Types

5.2.2 GetFeature

In this implementation the GetFeature request performs the following main steps [1]

1. Mapping the Feature Type Name to layers
2. Fetching the Geometries and other data from database[1]
3. Getting the Bounding Box of the received geometries[1]
4. Converting the data to internal format[1]
5. Converting the data to XML format[1]
6. Selecting the proper objects from the XML and adding them to the WFS response frame

According to the tests, four of these, marked with[1], cause significant delays in the operation. The step 4, where data is converted from geometries and other properties to the internal data format is quadratical, with around O(N^2) efficiency. The other steps are linear with O(N) efficiency. With small datasets, the conversion from internal format to the XML format (step 5) takes the most time. The time consumption depends of the amount of geometries that are delivered from the database to the WFS server, not the amount of total geometries in the database. With datasets larger than some 18 000 geometries to convert, the step 4 takes the most time.

It's worth noting that the physical limitations of the test server, may cause the time consumption to increase even faster with bigger datasets, than it does with small ones. For example in case where memory is getting low, the computer starts paging [23] memory to hard drive. Hard drive operations are very slow, compared to memory operations, and thus the need to page will slow down the whole process.

Without Filtering

Requesting data without filtering is the basic case for comparing the performance. In this case, all data of certain Feature Type is fetched and returned. From user's point of view, there is no filtering. In this implementation, there is still some filtering from database to WFS response in two stages:

[1]These steps cause significant delays in the Get Feature operation and they are shown in the performance graphs.

1. Only data from certain layers is fetched
2. Only the data that gets mapped to the asked feature type is added to the WFS response frame

These steps are explained in more detail in 4.3.2. It is good to remember that the time consumption of the quadratical efficiency operation depends on the amount of data that is delivered to WFS server after the filtering in step 1. As a consequence, further filtering at step 2 will not speed up that operation.

The tests for request without filtering were performed with Firefox on remote computer with URL <`http://host/service.aspx?service=WFS&version=1.1.0&request=GetFeature&TYPENAME=kanta:Rakennus`>.

The Figure 5.3 shows the measurements as graph.

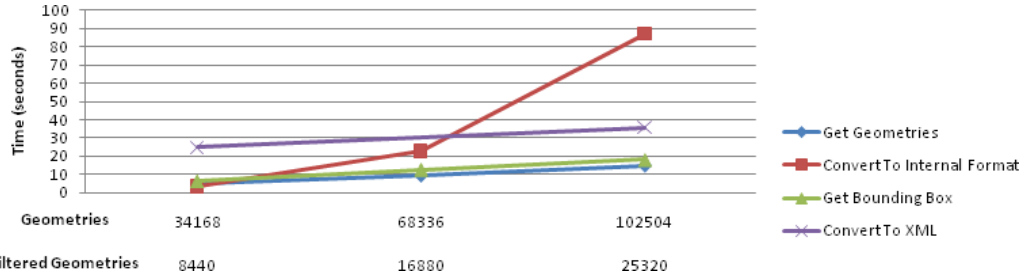

Figure 5.3: Basic Database: The Performance of Get Feature Operation Without Filtering

Bounding Box Filtering

The tests for request with Bounding Box filtering were performed with Firefox on remote computer with URL <`http://host/service.aspx?service=WFS&version=1.1.0&request=GetFeature&TYPENAME=kanta:Rakennus&BBOX=2503997.8081,6681999.78,2504100.8081,6682250.78`>.

The Figure 5.4 shows the measurements as graph.

Property Filtering

With the Property Filtering it is very clear that the performance is dependent on the amount of geometries that WFS server fetches from the database.

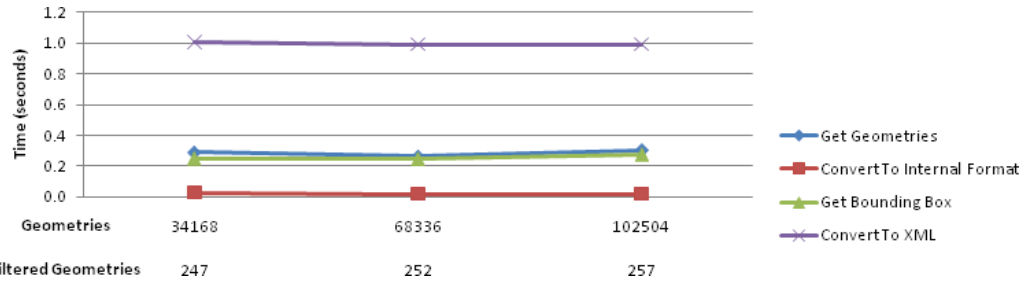

Figure 5.4: Basic Database: The Performance of Get Feature Operation With Bounding Box Filtering

With the Basic Database, the server has to fetch and convert all the geometries with the requested feature type. The total amount of converted geometries will likely be much larger than the amount of the filtered geometries in the end result. This is also the case in these measurements: The unfiltered operation that results in 25 320 geometries takes about the same amount of time than the Property Filtering for the same feature type that results in only 8 208 filtered geometries.

The Property Filtering was tested with Gaia on remote computer. Gaia was used to filter with parameters "kanta:RakennuksenKayttotarkoitus IsEqualTo 'asuinrakennus' ". Wireshark was used to find out the returned geometry count from the WFS response header.

The Figure 5.5 shows the measurements as graph.

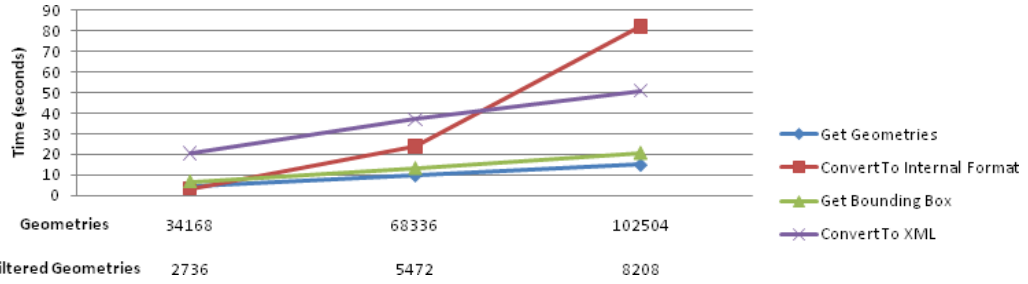

Figure 5.5: Basic Database: The Performance of Get Feature Operation With Property Filtering

5.3 The Improved Database

These measurements are done after I implemented the improvements described in Section 4.3.3. There was a huge improvement on the performance of the GetCapabilities operation. Also the performance of GetFeatures operation when using the Property Filtering improved greatly. With these changes it is also possible to use the Maximum Features limit to keep the response times small enough in all the usual tasks.

Note that the tests on Improved Database were run with dataset with up to 130 000 geometries while the tests with Basic Database were only run with up to 100 000 geometries.

5.3.1 GetCapabilities

With the changes, there is only one very fast step in the process: the available Feature Types and their Bounding Boxes are fetched as strings from the database table. The bounds have been pre-calculated there after inserting the data to the database. The time consumption seems to be around constant 22ms. The time consumption shouldn't be related to amount of geometries and according to the measurements this is true.

5.3.2 GetFeature

Without Filtering

The performance of without filtering remained unaffected after the improvements. This was expected because the improvements were made to filtering performance. When comparing the graph to the graph of Basic Database, remember the additional datapoint of 130 000 geometries. The Figure 5.6 shows the measurements as graph.

Bounding Box Filtering

The performance of Bounding Box Filtering also remained unaffected after the improvements. This was also expected because there were no changes to the spatial filtering functionalities. The both implementations use Oracle Locator's spatial query functionalities which, with proper spatial indexes in

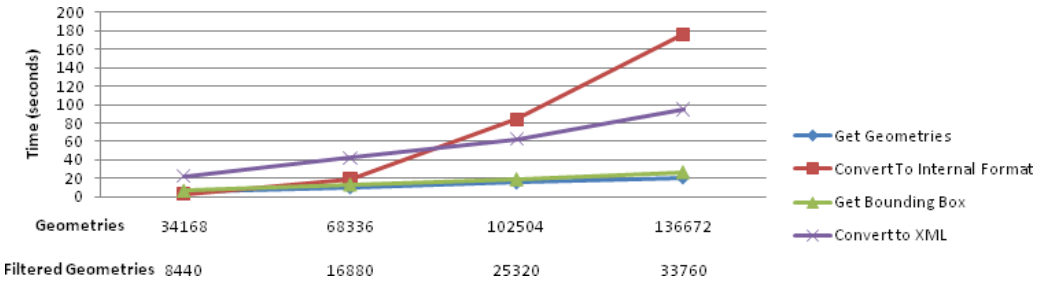

Figure 5.6: Improved Database: The Performance of Get Feature Operation Without Filtering

database, seem very efficient. The Figure 5.7 shows the measurements as graph.

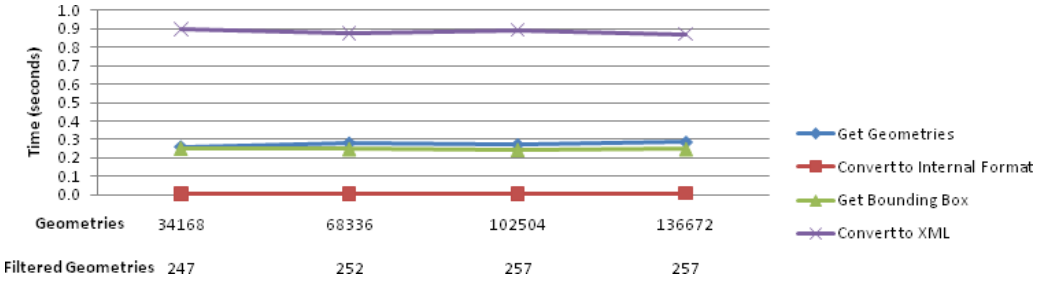

Figure 5.7: Improved Database: The Performance of Get Feature Operation With Bounding Box Filtering

Property Filtering

In the Property Filtering there is a clear improvement in performance. The time for fetching 8000 filtered geometries from 100 000 total geometries fell from around 3 minutes to 30 seconds. The improvement makes more difference if there are more total geometries and fewer geometries get filtered. The time for completing the request is dependent mostly on the count of filtered geometries, not so much on the total geometries. With Basic Database the case was just the opposite. The Figure 5.8 shows the measurements as graph.

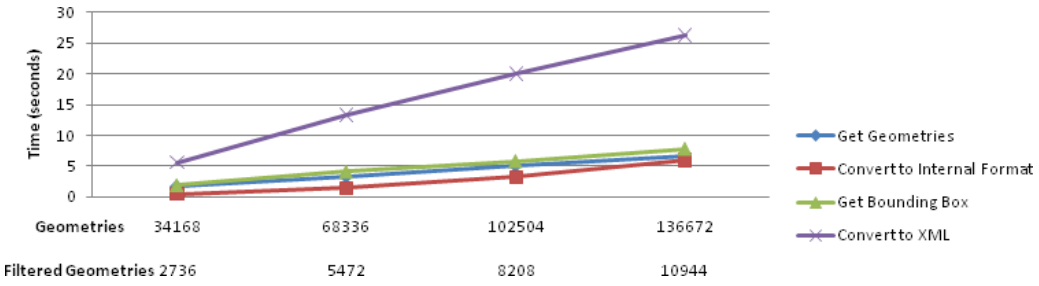

Figure 5.8: Improved Database: The Performance of Get Feature Operation With Property Filtering

Chapter 6

Conclusion

In this chapter I will shortly describe the most important findings of the study. I will also specify the fields where further research could be done to obtain improved performance.

6.1 Main Conclusions

Converting data from one data format to another without data loss can be difficult because of the differences in the allowed data. Using Object Data to store properties offers a lot of potential for reducing the data loss when converting data from XML to CAD drawing and back. For hierarchal data, generated XData can be used to, at least temporarily, define the hierarchy relations in a CAD drawing.

An Oracle database can be used to store spatial data with attributes. This data can be efficiently filtered with spatial and attribute queries. The attributes from a XML file can be serialized into a single field in the database. This serialized field can be queried by using the LIKE-operation. The performance is easily good enough to be usable; in the implementation done in this study, the delays from the LIKE-operation were very small compared to the other delays in the processing of a WFS request.

The WFS response is easy to build from a XML file that has already been filtered with the methods above. Building the response is basically just selecting nodes from the XML file and adding them to the WFS response frame.

For WMS server it is probably easiest to use software that is already available on market, like MapGuide Open Source. The WMS server can use the database or the XML file can be converted to CAD drawing and then exported to WMS.

6.2 Further Research

The performance of the attribute query from the Oracle database should be researched further. At the database level, it would probably be the first thing that starts to slow down the filtering process. One could probably find some way to serialize the attributes from the XML in such a way that Oracle could use its indexing optimizations. With large amount of entities and attributes this would make quite a difference.

On the area of mapping there would be work to be done in finding the best method for mapping hierarchal data. In this thesis the issue was approached theoretically, from the perspective of what is possible and what is not. The performance point-of-view for this was left out of the scope. In the described implementation, the algorithms that cause the $O(N^2)$ complexity in one of the steps, are related to this issue. At the moment, in typical use, this is not a problem, but it is something that will likely come up some time in the future.

Appendix A

Appendix

A.1 Test Records

The test records are available on request. They are too large to fit into this document but digital delivery is not a problem.

A.1.1 Get Capabilities Measurements Spreadsheet - tr_get_capabilities.xls

A.1.2 Get Features Measurements Spreadsheet - tr_get_features.xls

A.2 Examples

A.2.1 Gaia WFS Client Search Filter

```
<ogc:And xmlns:ogc="http://www.opengis.net/ogc">
    <ogc:PropertyIsEqualTo matchCase="true">
        <ogc:PropertyName>tyyppi</ogc:PropertyName>
        <ogc:Literal>seinälinja</ogc:Literal>
    </ogc:PropertyIsEqualTo>
    <ogc:PropertyIsEqualTo matchCase="true">
        <ogc:PropertyName>rakennuksenKayttotarkoitus</ogc:PropertyName>
        <ogc:Literal>asuinrakennus</ogc:Literal>
    </ogc:PropertyIsEqualTo>
```

</ogc:And>

Bibliography

[1] Autodesk. Importing/Exporting supported vector formats, Autodesk Map 3D 2007. <http://usa.autodesk.com/adsk/servlet/ps/item?siteID=123112&id=7146334&linkID=9240857>, 19 April 2006.

[2] Elmagarmid, A. K., Ed. *Database transaction models for advanced applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[3] ESRI. WMS Connector for ArcIMS 9.2. <http://webhelp.esri.com/arcims/9.2/general/mergedProjects/wms_connect/wms_connector/get_capabilities.htm>, 12 May 2009.

[4] European Parliament. Infrastructure for Spatial Information in the European Community (INSPIRE) Directive. <http://inspire.jrc.ec.europa.eu/>, 24 March 2009.

[5] Herva, V. Using delaunay triangulation in infrastructure design software. Master's thesis, Helsinki University of Technology, 28 January 2009.

[6] JUHTA Julkisen hallinnon tietohallinnon neuvottelukunta, JHS-suositukset. JHS 154 ETRS89-järjestelmään liittyvät karttaprojektiot, tasokoordinaatistot ja karttalehtijako, Liite 5: Muunnos ykj <-> ETRS-TM35FIN. <http://docs.jhs-suositukset.fi/jhs-suositukset/JHS154_liite5/JHS154_liite5.html>, 24 March 2009.

[7] JUHTA Julkisen hallinnon tietohallinnon neuvottelukunta, JHS-suositukset. JHS 162 - Paikkatietojen mallintaminen tiedonsiirtoa varten (Modeling Spatial Data for Transfer). <http://www.jhs-suositukset.fi/web/guest/jhs/recommendations/abstracts#162_240>, 24 March 2009.

[8] Kangasharju, J., and Tarkoma, S. Benefits of alternate xml serialization formats in scientific computing. In *SOCP '07: Proceedings of the 2007 workshop on Service-oriented computing performance: aspects, issues, and approaches* (New York, NY, USA, 2007), ACM, pp. 23–30.

[9] Kirzner, I. M., Ed. *The Irresistible Force of Market Competition*, vol. 50. The Foundation for Economic Education, March 2000.

[10] Norman Walsh, O'Reilly XML.com. XML Schema Web Site. <http://www.xml.com/pub/a/1999/07/schemas/validity.html>, 01 July 1999.

[11] Open Geospatial Consortium, Inc. OpenGIS Web Feature Service (WFS) Implementation Specification 1.0, OGC 02-058. <http://portal.opengeospatial.org/files/?artifact_id=7176>, 17 May 2002.

[12] Open Geospatial Consortium, Inc. OpenGIS Web Feature Service (WFS) Implementation Specification 1.1, OGC 04-094. <http://portal.opengeospatial.org/files/?artifact_id=8339>, 03 May 2005.

[13] Open Geospatial Consortium, Inc. OpenGIS Web Map Server Implementation Specification 1.3, OGC 06-042. <http://portal.opengeospatial.org/files/?artifact_id=14416>, 15 March 2006.

[14] Open Geospatial Consortium, Inc. OpenGIS Geography Markup Language (GML) Encoding Standard Web Site. <http://www.opengeospatial.org/standards/gml>, 03 March 2009.

[15] Open Geospatial Consortium, Inc. OpenGIS Web Feature Service (WFS) Implementation Specification Web Site. <http://www.opengeospatial.org/standards/wfs>, 09 March 2009.

[16] Open Geospatial Consortium, Inc. OpenGIS Web Feature Service (WFS) schema version 1.0.0. <http://schemas.opengis.net/wfs/1.0.0/WFS-basic.xsd>, 13 October 2009.

[17] Open Geospatial Consortium, Inc. OpenGIS Web Map Service (WMS) Implementation Specification Web Site, 03 March 2009.

[18] Oracle. Whitepaper: Oracle Locator: Location-Enabling Every Oracle Database. <http://www.oracle.com/technology/products/spatial/pdf/10gr2_collateral/locator_twp_10gr2.pdf>, August 2005.

[19] RAND Worldwide. AutoCAD Map 3D 2010, Features and Benefits. <http://www.rand.com/imaginit/1/pdfs/technology/software/autocadmap3d2010_features-benefits_final.pdf>, 01 March 2010.

[20] Shanmugasundaram, J., Shekita, E., Barr, R., Carey, M., Lindsay, B., Pirahesh, H., and Reinwald, B. Efficiently publishing relational data as xml documents. *The VLDB Journal 10*, 2-3 (2001), 133–154.

[21] Shekhar, S., Vatsavai, R. R., Sahay, N., Burk, T. E., and Lime, S. Wms and gml based interoperable web mapping system. In *GIS '01: Proceedings of the 9th ACM international symposium on Advances in geographic information systems* (New York, NY, USA, 2001), ACM, pp. 106–111.

[22] Stefan A. Voser, MapRef.org. A very short introduction to Coordinate Reference Systems. <http://www.mapref.org/AveryshortintroductiontoCoordinateRefere.html>, 01 March 2010.

[23] Webopedia Computer Dictionary. What is paging. <http://www.webopedia.com/TERM/P/paging.html>, 24 March 2009.

[24] Webopedia Computer Dictionary. What is spatial data. <http://www.webopedia.com/TERM/S/spatial_data.html>, 24 March 2009.

[25] Wei, Z.-K., Oh, Y.-H., Lee, J.-D., Kim, J.-H., Park, D.-S., Lee, Y.-G., and Bae, H.-Y. Efficient spatial data transmission in web-based gis. In *WIDM '99: Proceedings of the 2nd international workshop on Web information and data management* (New York, NY, USA, 1999), ACM, pp. 38–42.

[26] World Wide Web Consortium (W3C). Extensible Markup Language (XML) Web Site. <http://www.w3.org/XML/>, 09 March 2009.

[27] World Wide Web Consortium (W3C). XML Schema Web Site. <http://www.w3.org/XML/Schema>, 09 March 2009.