Jari Peuralahti

Geographic Information System

- A Case Study for Developers

Helsinki Metropolia University of Applied Sciences

Master of Engineering

Degree Programme in Information Technology

4 February 2014

Author Title Number of Pages Date	Jari Peuralahti Geographic Information System – A Case Study for Developers 100 pages + 3 appendices 4 Feb 2014
Degree	Master of Engineering
Degree Programme	Information Technology
Specialisation option	Media Engineering
Instructor	Kari Salo, Principal Lecturer

Geographic Information Systems, also abbreviated as GISs, are systems or application packages that have been used traditionally by cartographers and geoinformation analysts. These digital systems are used to manage geographical information processing or processes to visualize geospatial information. These systems can also be used to manage maps, or precisely, digital maps.

This master's thesis aimed to resolve what GIS is or means. The theory part describes, on a general level, the background, technologies and standards of GIS as described in the literature. The empirical part illustrates practical examples, how a GIS and a Web-based map application can be implemented by using open source solutions.

This study aimed to summarize the essential aspects that the uninitiated application or system developer should understand about a geographical information system and its role, when creating geospatial applications or integrating them to be a part of other information systems.

The results of the thesis, showed that the GIS seems more complex than it is, due to the development history and the field of science where it has mainly used. From a developer's point of view, the most significant finding is, that the geographical information system requires the collection of software, hardware, information, and people. Without this set of collection, the GIS cannot be implemented. From a technological point of view, the GIS is like other information systems. However, its unique feature is effectively to handle the information with a geometrical form and a place in the spatial space. Therefore, in general, the goal of any GIS is to visualize and manage geospatial data over the maps. In practice, the most important component of the system is a geoinformation server. On an abstract level, this server can be seen as a publication system for the spatial data sources, as well as a messaging broker, which mediates spatial data between applications and data storages. Geographic information systems have evolved over the years, from independent systems towards open and standardized Web service architecture. The GIS technology and related protocols and services have been well standardized by the Open Geospatial Consortium.

Keywords	Geographic Information System, GIS, Web GIS, OGC, OpenLayers, GeoServer, Google Maps

Tekijä Otsikko Sivumäärä Aika	Jari Peuralahti Paikkatietojärjestelmä – Tapaustutkimus kehittäjille 100 sivua + 3 liitettä 04.02.2014
Tutkinto	insinööri (YAMK)
Koulutusohjelma	tieto- ja viestintäteknologia
Suuntautumisvaihtoehto	mediatekniikka
Ohjaaja	yliopettaja Kari Salo

Paikkatietojärjestelmät ovat järjestelmiä tai sovelluskokonaisuuksia, joita on hyödynnetty perinteisesti karttapiirtäjien ja paikkatietoanalyytikkojen keskuudessa. Näiden digitaalisten järjestelmien avulla voidaan hallita esimerkiksi maantieteellisten tietojen käsittelyä tai niiden visualisointiin liittyvää prosessointia. Järjestelmillä voidaan myös hallita myös karttoja, tai tarkemmin sanottuna digitaalisia karttoja, joiden päälle on visualisoitu paikkatietopohjaista informaatiota.

Opinnäytetyössä selvitetään, mikä ja mitä paikkatietojärjestelmä on. Työn teoriaosuudessa käydään yleisellä tasolla läpi niitä taustoja, teknologioita ja standardeja joita paikkatietojärjestelmäteknologiasta on alan kirjallisuudessa kerrottu. Empiirinen osuus havainnollistaa käytännön esimerkkien kautta, miten avoimen lähdekoodin ratkaisuja hyödyntämällä voidaan toteuttaa paikkatietojärjestelmä sekä sitä hyödyntävä www-pohjainen karttasovellus.

Lopputyö tiivistää ne olennaiset asiat, jotka asiaan vihkiytymättömän sovellus- tai järjestelmäkehittäjän tulisi ymmärtää paikkatietojärjestelmästä ja sen merkityksestä osana paikkatietosovelluksia tai muita tietojärjestelmiä.

Työn lopputuloksena voidaan todeta, että paikkatietojärjestelmä vaikuttaa monimutkaisemmalta kuin se on, mikä johtuu sen kehityshistoriasta ja tieteenalasta, jossa sitä pääsääntöisesti on hyödynnetty. Kehittäjän näkökulmasta merkittävin havainto on, että paikkatietojärjestelmä edellyttää kokoelman ohjelmistoja, laitteita, tietoa ja ihmisiä. Ilman tätä edellytystä paikkatietojärjestelmää ei voida toteuttaa. Teknologisesti paikkatietojärjestelmä ei poikkea juurikaan muista tietojärjestelmistä muuten, kuin että sillä voidaan käsitellä tehokkaasti tietoa, jolla on myös geometrinen muoto ja tila-avaruuteen sijoitettu paikka. Käytännössä järjestelmän keskeisin komponentti on paikkatietopalvelin, jonka voi nähdä julkaisujärjestelmänä paikkatietovarastojen tiedoille sekä sovelluksien ja tietovarastojen välisen sanomaliikenteen välittäjänä. Paikkatietojärjestelmät ovat kehittyneet ajan saatossa itsenäisistä järjestelmistä tukemaan hajautettua verkkopalveluarkkitehtuuria. Paikkatietoteknologia ja siihen liittyvät yhteyskäytännöt sekä palvelut on hyvin standardisoitu Open Geospatial Consortiumin toimesta.

Avainsanat	paikkatietojärjestelmä, GIS, Web GIS, OGC, OpenLayers, GeoServer, Google Maps

Contents

1 Introduction

1.1 Background to the Study

The subject for this study emerged from a development project, where the Finnish Communication Regulation Authority (FICORA) was developing systems and processes to improve situation awareness for the event and disturbance management. One of the project goals was to develop a disturbance map application, to publish Finnish telecommunication operators' major communication network failures for public, in near real time. At an early state of the project, it was discovered that the disturbance map application needs a system which can handle geographical information to help the process to illustrate the failure areas on the map.

However, from a development point of view, the keywords "geographical information" and "maps" raised many questions to solve before the implementation of the map application project could be started. In the beginning, the common knowledge shared by other ICT engineers, at the house, was that there are commercial "geographical information systems" or "map engine systems" that are used by professionals such as cartographers, map engineers or analysts. However, all the questions related to the technology itself raised shoulders and were referred to ask an expert of these systems.

The area of the subject seemed to be too mystified even to get a straight answer to the simple questions, what a geographic information system is, and how it really works from the technology point of view. At this point, it seemed that this mystified area of technology had to be studied more to get the full picture and idea, how a simple Web mapping application should be done. Therefore, this master's thesis tries to demystify the question what a geographic information system is and what should a developer generally understand from this area of technology.

1.2 Goals and Limitations of the Study

The main goals of this thesis are: to investigate what is a geographic information system, to explore the key technologies, services and standards involved in this area of the subject and to create a simple Web map application to understand the programming aspect of the system.

To achieve these goals, it is necessary to have a theoretical view of the literature, to comprise geographical information systems and the related subjects. Therefore, the theoretical part includes compilation of the key findings, and discussion on the area of the thesis subject. The knowledge gained by the literature studies are used in the empirical part, with a focus on, how a geographical information system test environment is implemented, and how a Web map application using the test environment is programmed.

The main focus of this thesis is to demystify the question what a geographical information system is and to exemplify how a simple Web map application is created. Therefore, this thesis tries to frame the scope of the study in such a way that it will explain all the key findings and examples as generalized as possible. For example, this thesis does not include discussion on the technological differences between different geographical information systems, databases or between the programming languages or libraries one should use.

1.3 Structure of the Study

This thesis is constructed of two larger entities; the theoretical part and the empirical part. The theoretical part discusses the concept of geographical information system including sections on the history of the system's development and anatomy, the key terminologies, services, geospatial data, system packages, applications and the related standards.

The empirical part shows how an example geographical information system environment is utilized using open source components, and how a simple Web map application is programmed using an existing spatial data, and how a bit complex Web map application is programmed which can use and manipulate several spatial data sources.

2 Theoretical Part of the Study

2.1 Geographic Information System

Caitlin Demsey (2012) writes in her article at the GIS Lounge that the question, *what is a GIS*, is probably the most asked question presented to people working in the field and is probably the hardest to answer in a clear and succinct manner.

According to Demsey, a geographic information system (GIS) is a computer system, or a technological field (science) that incorporates geographical features with data. These geographical features are used in order to map, analyze, and assess real-world problems. Dempsey also states that the key word in GIS is geography, meaning that some portion of the data is contained spatial information referenced to some place on the Earth. (Caitlin Demsey, 2012.)

A GIS is also defined to be a system of hardware or software which is capable of capturing, storing, editing, manipulating, managing, analyzing and displaying geographically referenced location data (Fu & Sun, 2011), including the personnel and procedures needed to operate the system (USGS, 2007) and a system that helps users to answer questions and solve problems by visualizing data in many ways that reveal relationships, patterns, and trends in the form of maps, globes, reports, and charts (ESRI, 2013).

In the Web GIS book (Fu & Sun, 2011, p. 4) outlines:

> "Everything that happens, happens somewhere. Knowing "what" is "where" and "why" it is there, can be critically important for making decisions in personal life, as well as an organization. GIS is the technology as well as science for handling the "where" type of questions and for making intelligent decisions based on space and location."

Michael DeMers has depicted GIS even broader sense, presenting the GIS as a collective, where collective (figure 1.) is a collection of software, hardware, data, and people (DeMers, 2009).

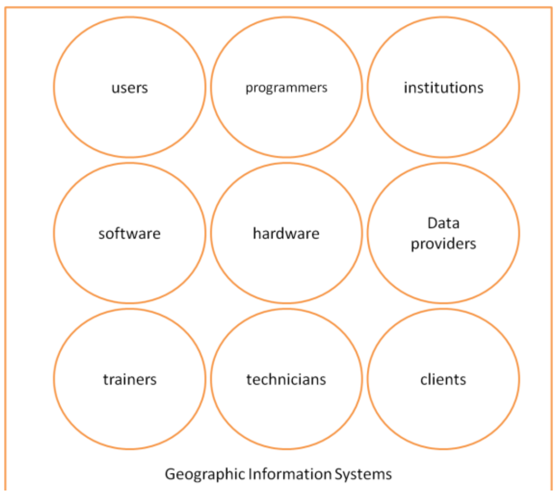


Figure 1: GIS collective. The GIS is a collection of software, hardware, data, and people. Reprinted from DeMers (2009)[11]

DeMers (2009) has listed the essential parts that make up the whole GIS as:

- Data and information
- Computers, input and output technology and software
- Geographic and related concepts that drive the analysis
- People, such as operators, managers, consultants, vendors, etc.
- Institutions and organizations within which the GIS exists” (DeMers 2009, p. 11).

In Miller's (2012) white paper, geodesign is defined as a method of creating or re-creating goods and services using geospatial information. The difference between ordinary design and geodesign is that data management and information analysis is based on geospatial information as illustrated in figure 2. The way how geodesign itself is performed depends on the company's needs to create new services or goods guided by the geospatial information (Miller, 2012).

Kraak and Ormeling (2010) have described a similar kind of model (as presented in figure 2) of a "problem solving production line" (Kraak & Ormeling 2010, 9). However, in this equivalent model, the terminology: *"GIS technology"," Manage Geo-Spatial Information"," Analysis" and "Geodesign"* are substituted by the terminology: *"GIS science", "Exploration", "Analysis" and "Synthesis and Presentation"*.

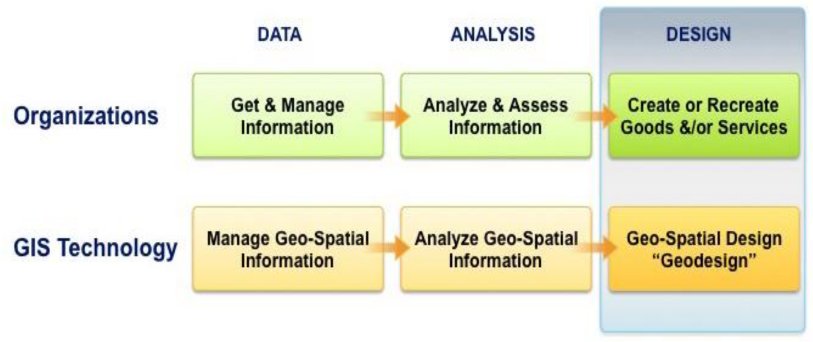

DATA
ANALYSIS
DESIGN
Organizations
Get & Manage Information
Analyze & Assess Information
Create or Recreate Goods &/or Services
GIS Technology
Manage Geo-Spatial Information
Analyze Geo-Spatial Information
Geo-Spatial Design "Geodesign"

Figure 2: The difference between design and geodesign. Reprinted from Miller (2012)[3]

Then, how good geodesign (or synthesis and presentation), geographic planning or decision making can be done, is based on the available data and analyzing tools available.

To simplify, the goal of any GIS (technology and science) is to do data visualizations, which are the key to the exploration to find an answer to questions such as "*What is the nature of the data set?*" or "*Which of those data sets reveals patterns related to the current problem studied?*" (Kraak & Ormeling, 2010).

In such a manner, data is the core of any GIS, and to be precise, spatial data coupled with attribute data. If the spatial data is referenced in some way to the position on the Earth, it is called as *geospatial* data. (Kraak & Ormeling, 2010.)

It is generally said that most of an organization's data include some reference to spatial data. For example in many presentations made by the spatial industry (ESRI, Microsoft, MapInfo, etc.), there is the repeated quote "*80% of all data includes some reference to spatial data (or geography)"* and these statistics have been passed down since the early 1990s. However, there is no real evidence this quote is correct, because no one seems to know where this argument originates from (as the original reference is missing). (Ball 2009; R.K 2012.)

In GISs, geospatial data is stored in databases which are grouped into two different types: vectors and rasters. The vector database includes spatial information in the form of lines, points or polygons (also named as *features*). The raster database contains cell-based pixel data such as aerial imagery, 2D maps, 3D maps or digital elevation models. (Fu & Sun 2011; Caitlin Demsey 2012.)

The geospatial and attribute data are created, edited, visualized and analyzed by the GIS applications, which are as a vital part of GIS technology as the data on exploration itself. There are many GIS applications available, each meant to perform some particular function(s). For example, a file format utility that converts spatial data from one type of GIS file into another, or a GIS application which helps serve data and interactive maps through the Internet. (Caitlin Demsey, 2012.)

To summarize, from all the above definitions, it could be said the GIS can be complex and diverse to understand as a whole. There is much variety in the terminology and abbreviations, as to where or how it is used, or what systems belong to it. All depends on the point of view or the content in with the GIS is reviewed. There has even been debate whether the GIS is a tool or science (Wright, et al., 1997).

Nevertheless, the GIS is the supporting science and technology for geodesign or visualization, which is a systematic methodology or an application for people who are doing (geographic) planning or decision making (Fu & Sun, 2011). The GIS has many areas involved, from technical components to process management and from data analyses to visual explorations. The learning curve to be a GIS professional in all these areas can be a very long and challenging journey, and to quote Caitlin:

> "It is essential to understand what kind professional skills are required in the GIS process, i.e. a person highly skilled in GIS analysis should not seek a job as a GIS developer" (Caitlin Demsey, 2012, p.4).

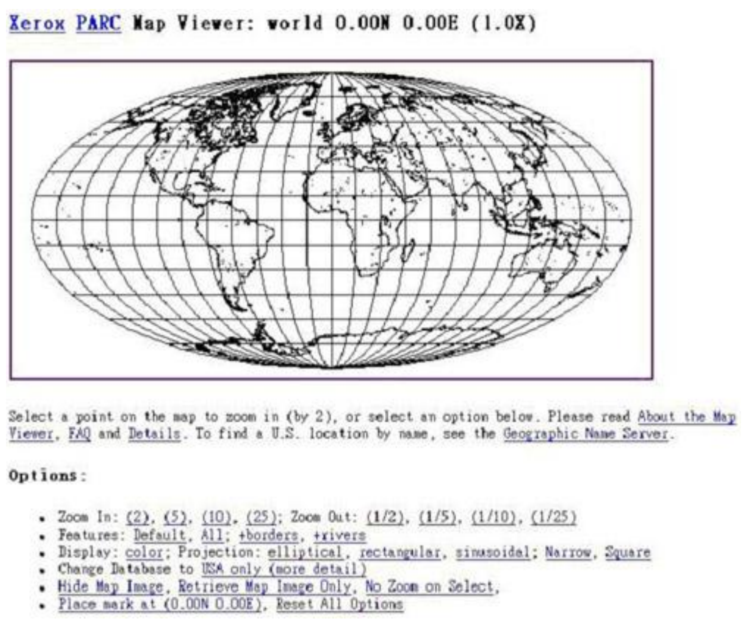
2.2 Brief History of the GIS

The history of the GIS is a field which is little more than anecdotal. There is little information or proven track record of where all this business and the resulting jobs have arisen. According to Coppock and Rhind (1991), computer-based GIS has been used since at least the late 1960s: their manual predecessors were in use perhaps 100 years earlier. The content of any history of the GIS depends in large on the definition adopted: is it a computer-based system for analyzing spatially referenced data or any system handling geographical data. Either way, the main background of those involved in the GIS development and history (and there have been several) have been in cartography, computer science, geography, surveying, commercial data processing, mathematics, statistics and remote sensing, and their goal of the developed systems have been in urban and regional planning, property owning ship, the management of utilities, military intelligence and tactics, taxation and many others. (Coppock & Rhind, 1991.)

However, the motivations for developing the GIS or elements of such a system have been in the desire to speed or intensify the conduct of operations on spatially referenced data. For example, Roger Tomlinson, working for Canada's Federal department of Forestry and Rural Development, faced the problem that it was impossible to analyze maps of East Africa at an acceptable cost, and a calculation made in 1965 prices indicated a need of 8 million dollars (Canadian) and the requirement for 556 technicians for three years, in order to overlay the 1:50 000 scale maps of the Canada Land Inventory. This unacceptable level of resource led Tomlinson to develop an automated method. It is generally adopted that Roger Tomlinson coined the term "GIS", and therefore he is recognized as the father of the GIS for his pioneering work developing the GIS and promoting GIS methods. (Coppock & Rhind 1991; Fu & Sun 2011.)

In addition to Tomlinson, Coppock & Rhind (1991) mention four other key persons who have had (and still have) a large influence on GIS development and techniques used today: Howard Fisher in the Harvard Laboratory for Computer Graphics (LCG), Jack Dangermond in the Environmental Systems Research Institute (ESRI) in North America, and David P. Bickmore at the Experimental Cartography Unit (ECU) in the United Kingdom. The most famous of the above names is ESRI's founder Jack Dangermond by the very reason that ESRI ArcGIS products are most widely adopted globally in various organizations around the world. Many of today's GIS de facto standards and techniques are derived from the development of the ESRI products, such as the vector-based system, Planning Information Overlay System (information layers over maps) and cell-based package GRID (raster image data file format). (Coppock & Rhind 1991; ESRI 2012.)

In the dawn of the World Wide Web, the Xerox Corporation Palo Alto Research Center (PARC) produced one of the first Web-based map viewer (named Map Viewer), in 1993, marking the origin of Web GIS. This first map viewer was an experiment in allowing retrieval of interactive information on the Web. It had single zoom capabilities, layer selection and map projection functions (see figure 3). (PennState, College of Earth and Mineral Sciences 2009; Fu & Sun 2011.)



Xerox PARC Map Viewer: world 0.00N 0.00E (1.0X)

Select a point on the map to zoom in (by 2), or select an option below. Please read About the Map Viewer, FAQ and Details. To find a U.S. location by name, see the Geographic Name Server.

Options:

- Zoom In: (2), (5), (10), (25); Zoom Out: (1/2), (1/5), (1/10), (1/25)
- Features: Default, All; +borders, +rivers
- Display: color; Projection: elliptical, rectangular, sinusoidal; Narrow, Square
- Change Database to USA only (more detail)
- Hide Map Image, Retrieve Map Image Only, No Zoom on Select,
- Place mark at (0.00N 0.00E), Reset All Options

Figure 3: Xerox Map Viewer. Reprinted from PennState, College of Earth and Mineral Sciences (2009)

The Xerox PARC's Map Viewer pioneered the method of running the GIS inside a Web browser.

Two years later, in 1995, a researcher at UC-Berkeley developed GRASSLinks, which is a Web interface top of GRASS (Geographic Resources Analysis Support System) GIS, an open source GIS package originally developed by the U.S. Army Construction Engineering Research Laboratories. GRASS is (GRASS 7) used currently in academic and commercial settings around the world, as well as in various governmental agencies including the National Space Agency (NASA), the National Oceanic and Atmospheric Administration (NOAA, USA), U.S Department of Agriculture (USDA), Docklands Light Railway (DLR, Transport of London), the Commonwealth Scientific and Industrial Research Organization (CSIRO, Australia), the National Park Service, the U.S. Census Bureau, U.S Geological Survey (USGS), and in various environmental consulting companies. (OSGEO, 1998-2013.)

In 1996, MapQuest cornered the market in providing turn-by-turn driving directions (navigation applications) in much the same way that Google is the most popular Web search engine today.

In 1997, the U.S. Census Bureau developed a Web interface for its enormously rich TIGER (Topologically Integrated Geographical Encoding and Referencing) (DeMers,

2009) dataset called the TIGER Map Server. The TIGER Map Server made it possible to toggle on/off many of the geographic entities in the dataset.

It is said that during the time from 1995 through 2005, the Web GIS sites represented the state-of-the art but suffered from two main flaws; complicated user interfaces and slow performance.

The next significant development in Web mapping occurred in 2005, when a new technology called AJAX (Asynchronous JavaScript and XML) enabled Web developers to create sites that corresponded desktop applications (dynamic Web page interactions). The year 2005 is also significant because this is the very year when Google released Google Maps and Google Maps API (Application Programming Interface) for Web developers (Google Inc., 2012). The result was an explosion of custom mapping applications and the addition of a new term for the Web mapping lexicon — *mashup*. (PennState, College of Earth and Mineral Sciences, 2009.)

> "The term mashup traces its roots to the music industry, where it is used to describe the mixing of tracks from two or more songs to produce a new song. Thus, the 3rd generation of Web mapping can be thought of as the 'mashup generation.'" (PennState, College of Earth and Mineral Sciences, 2009)

Shortly after the launch of Google Maps, other companies followed Google's example by publishing their own Web map applications, such as Microsoft Corp. (Microsoft Virtual Earth), Yahoo! (Yahoo! MAPS), and America Online (MapQuest). Worth mentioning is that the data used by Google and MapQuest in their online maps comes from NAVTEQ, a Chicago-based company which was acquired in 2008 by Nokia.(NAVTEQ Maps, 2012.)

Geographic information systems today have evolved considerably since the term GIS was coined, from stand-alone-one-purpose-analyze-applications to an open-cloud-Web-for-everyone-applications, where the power of GIS tools and functions has been exposed to the Internet community so that there are no need of long education curve to become a GIS expert, or even understand what the GIS is, to make a GIS application. So, many of the Web developers today, creating a Web map application using, for example, Google Map APIs, do not really know they are working on an area called GIS.

2.3 GIS Terminology

Fu & Sun (2011) have defined four different GIS terms commonly used by people and their relations to each other (see figure 4): GIS, distributed GIS, Internet GIS and Web GIS.

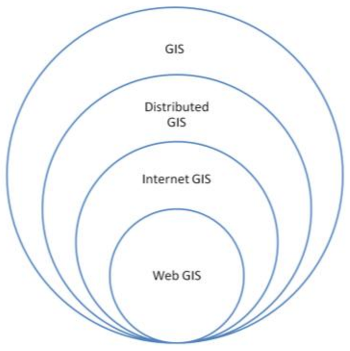


Figure 4: GIS terms commonly used by people. Reprinted from Fu & Sun (2011)[14]

The term GIS covers all the flavors of GIS but does not define do the GIS has a property of distributed GIS, Internet GIS or Web GIS. The distributed GIS defines that GIS components, data and application usage, may be distributed in the company's local area network (LAN), or a wide area network (WAN), but are not exposed to the Internet.

Internet GIS is the system that uses any of the Internet services, and is thus theoretically broader than the Web GIS, which is the chief attraction of the Internet and is the most commonly used Internet service. Therefore, the Web GIS is the most pervasive form of the Internet GIS. The Internet GIS and the Web GIS are often used synonymously. However, the two are slightly different. The Internet supports many services, and the Web is only one of them. The Web GIS is also closely related to the term geospatial Web (or GeoWeb). The one definition for geospatial Web is that geospatial information is merged with abstract (no geospatial) information, such as Web pages, videos, photos and news. The definition is also closely linked to the geotagging and geoparsing research areas of the Web GIS. (Fu & Sun, 2011.)

2.4 Web Service Applications

Basic Web service applications usually have a three-tier architecture (or multitier architecture) as illustrated in figure 5, where these three layers are data tier, logical tier and presentation tier.

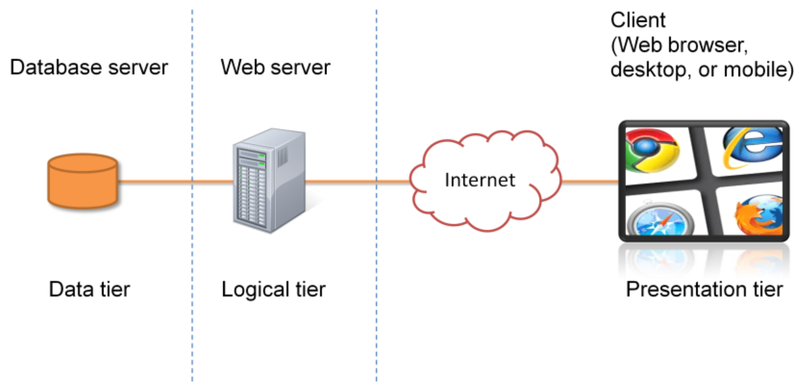


Figure 5: Three-tier architecture. Modified from Fu & Sun (2011)[26]

The data layer provides access to external systems such as databases incorporating all the application data stored. The logical tier implements the functionalities required by the application, such as the Web services used by the client application. The logical tier is also called a business tier or middle tier.

A fundamental rule in the three tier architecture is that the client tier never communicates directly with the data tier and all communication must pass through the middle tier. Conceptually the three-tier architecture is linear. In the three-tier model, the logical tier usually has the Web servers hosting the Web sites that read and serve the data for other Web applications to use. Web servers are responsible for accepting requests from clients and serving clients with responses. Web servers are also containers that allow certain scripts such as JSP, Perl, PHP, Java Servlet, and ASP. NET to run inside. Such scripts allow Web developers to perform the certain business logic. Some examples of Web application servers are Apache Tomcat, Microsoft Internet Information Services for Windows (IIS). (Microsoft 2013; Papagelis 2013.)

The presentation tier incorporates Web browsers, which are software applications for retrieving and presenting information resources on the Web. They enable a method of looking at and interacting with the World Wide Web (www). For most people, Web browser represents the "face" of the Web. However, technically, a Web browser is a client application that implements Hyper Text Transfer Protocol (HTTP) specifications, Hyper Text Markup Language (HTML), and JavaScript specifications, meaning that the browser knows how to communicate with Web servers, how to display an HTML page, and how to interpret and execute the JavaScript application code. The most popular Web browsers are Mozilla Firefox, Internet Explorer, Opera, Apple Safari and Google Chrome. There are slight differences in how these browsers support HTML and JavaScript specification standards led by the World Wide Web Consortium (W3C), meaning that the same Web application may appear and behave differently in different Web browsers. (Fu & Sun 2011; W3C 2012.)

The data tier incorporates database management systems (DBMS), which are a set of software programs, such as Microsoft SQL or PostgreSQL, that allows Web server applications (through Web browsers commanded by users) to create, edit and update data in database files, and store and retrieve data from those database files. The Web server application mediates data between Web server and database server using DBMS specific Structured Query Language (SQL) specifications over TCP/IP connection. (Connolly & Begg, 2005.)

To simplify, the basic architecture of Web application service is a compound of the client, Web server and database software, which serves in conjunction the intended Web service for the user. The client is a Web browser running the service application hosted by a Web server. The service application data are managed by the service application, and stored in the database server. A user of the Web service can access and consume the data through the Web server hosting the Web service application. The client, Web server and database software are thus logically separated. However, all the software components may yet run physically in the same machine but in security, performance and modularity reasons they should be physically separated to run in own machines.

As the client technology has evolved in the past years by drastically, gaining more processing power and memory, the trend seems to be that even more service processing and data storages are transferred to the client end (see section 2.7). However, even within these cases, the data, logical and presentation tiers do always exists, yet not so much tied to the physical architecture model.

Web GIS

The Web GIS is a combination of the Web server and GIS server software. The simplest form of the Web GIS should have at least a Web server and a client, where the Web server is loaded with some geospatial services to gain the ability to manipulate geospatial data. The goal of geospatial services is to help the developer to create applications which handle geospatial data in some form.

Geospatial services enabled by the GIS server software can be categorized by the functions they provide:

1. Map Services,
2. Data Services,
3. Analytical Services, and
4. Metadata Catalog Services.

However, for the capacity, quality and modularity reasons the GIS server and Web server software should be divided as illustrated in figure 6, offering the above geospatial services as Web services for Web GIS applications.

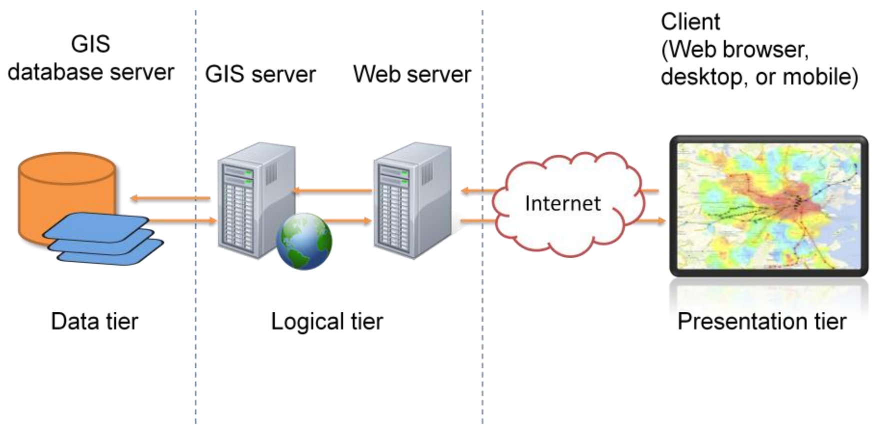


Figure 6: Three-tier architecture and GIS Server. GIS server provides geospatial services for Web applications and is usually implemented in same logical tier as Web servers. Modified from Fu & Sun (2011)[33]

The Need for Web Services

According to Fu & Sun (2011), an explosion of Web-based mapping applications followed the birth of the Web GIS and a flock of GIS software products, with a Web application user interface, appearing on the market. These early “Web-enabled” GIS technologies had several limitations, such as their internal architecture (isolated systems) and their ability to integrate with other information systems (not exposing the programming interfaces to other IT systems). Because of these limitations, the Web GIS was underused, and its potential was not realized. (Fu & Sun, 2011.)

Huang (2002) has stated that there are problems behind isolated systems:

> “Systems that do not expose programming interfaces cannot communicate with each other. In cases where functions and information in one system are needed by another, information cannot be shared when neither system can call the other” - (Huang, 2002)

One solution for exposing the isolated systems programming interfaces was to develop Web service technology.

According to W3C, the Web service is a Web application which can publish its services, service functions or messages on the Internet. The contemporary Web technology prefers the data format used for exchanging messages between Web servers and clients are XML or JSON (JavaScript Object Notation). The transfer protocol for XML and JSON is HTTP (Hypertext Transfer Protocol). (W3C, 1999 – 2013.)

The definition of Web service has evolved over the years. Previous definitions of Web service were mostly tied to XML (eXtended Markup Language), WSDL (Web Service Description Language), and SOAP (Simple Object Access Protocol). With the considerable changes that have been made in the Web service technology, SOAP is not the only way to implement Web services anymore. REST-style (Representational State Transfer) Web service has expanded the Web service concept into a more inclusive definition:

> "A Web service is a program that runs on a Web server and exposes programming interface to other programs on the Web." (Fu & Sun, 2011).

Therefore, from the above definitions the Web-enabled GIS server should offer the geospatial services as Web services, and communication between the GIS server, Web server and Web client are performed by using XML+HTTP or JSON+HTTP specifications.

Fu & Sun (2011) have described that the Web-enabled GIS server is named as "*Web GIS Server*":

> "The Web GIS server is the most important component in a Web GIS. Its functionality, ability to be customized, scalability, and performance are critical to the success of the Web GIS applications. The capability and quality of a Web GIS application are largely determined by the Web GIS server it uses." (Fu & Sun, 2011).

2.5 Web Services of GIS

As mentioned in section 2.3 the Web GIS is a combination of a Web server and GIS server software and the role of GIS server is to help developers to build applications which handle geospatial (or geographically referenced) data in some form.

The power of GIS server software is that it can produce geospatial Web services for an application developer. The geospatial services can be categorized by the functions they provide: Map Services, Data Services, Analytical Services, and Metadata catalog services, as explained below.

Web Map Services

Map services enable clients to request maps for a particular geographic extent (see section 3.2), and the maps are returned in an image format. The Map Service is the most general type of geospatial Web service.

The Open Geo Spatial Consortium (OGC) have specified that Map Service is called Web Map Service (WMS) and the service interface standard provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases. A WMS request defines the geographic layer(s) and area of interest to be processed. The response to the request is one or more geo-registered map images (returned as JPEG, PNG, GIF, etc.) that can be displayed in the client application. The interface also supports the ability to specify whether the returned images should be transparent so that layers from multiple servers can be combined or not. (OGC, 1994 – 2013.)

The OGC WMS standard has also defined that a “map” is the portrayal of geographic information as a digital image file format suitable for display on a computer screen (Open Geospatial Consortium Inc., 2006, p.5).

An example below illustrates the WMS request to a GIS server made by a client using REST-full query (Open Geospatial Consortium Inc., 2006, p. 77):

```
http://a-map-co.com/mapserver.cgi?VERSION=1.3.0&REQUEST=GetMap&
CRS=CRS:84&BBOX=-97.105,24.913,-78.794,36.358&
WIDTH=560&HEIGHT=350&LAYERS=AVHRR-09-27&STYLES=&
FORMAT=image/png&EXCEPTIONS=INIMAGE
```

Where the HTTP requests encompass the GIS server address, map service name and request parameters for the GIS service to reply. The map service processes this request and returns the requested image back (in PNG format) to the client, which is in this case example, a hurricane picture of the Gulf of Mexico, as illustrated in figure 7.

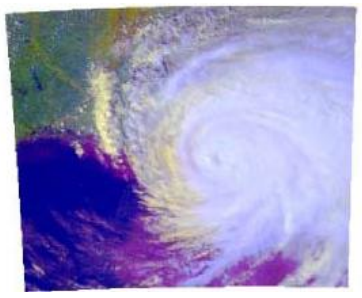


Figure 7: WMS image of the Gulf of Mexico. Reprinted from OGC 06-042 specification (2006)[77]

Beyond the mapping and map viewing, GIS services may also support the attribute query spatial identify, and dynamic re-projection functions. The attribute query is a request for records of features in a table based on their attribute values (ESRI Support, 2013). The dynamic re-projection means that GIS server converts requested geometry information from one coordinate system to another.

Map services can either be dynamic or cached. A map service that fulfills requests with pre-created tiles from the cache is called a cached or tiled maps service. The OGC specifies the Web Map Tile Service (WMTS) standard for this kind map service usage (OGC, 2010).

A cached map service can significantly improve performance time in delivering maps and is typically used to serve base maps (discussed in section 3.1) or maps where the content is relative static and changes very little over time. A dynamic map service requires the GIS server's to render the map each time a request comes into the GIS server. Dynamic map services are typically used to serve maps whose data is continually changing or to server maps with operational or theme layers, for example, weather maps.

Data Services

Data services allow Web clients to query, edit, and synchronize data over the Web. Some data services are also map services (as described earlier in this chapter), which let the Web client to see the map display as well as have access to raw data. (Fu & Sun, 2011.)

Web Feature Editing Service

A feature-editing service enables Web editing, which allow end-users to add, edit, and delete features in the geodatabases remotely via a Web GIS application. End-users can use a set of simple sketch tools to draw new features (polygons, lines, curves, rectangles etc.) or reshape existing features directly in a Web map produced by a Web application (Hazzard, 2011). This includes merging or splitting existing feature shapes. Users can also edit attributes and add attachments such as digital photos.

In the OGC's specifications (Open Geospatial Consortium Inc., 2005) the Web Feature Service (WFS) allows a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services.

According to the specification, the requirements for a Web Feature Service are the following:

1. The interfaces must be defined in XML.
2. GML must be used to express features within the interface.
3. At a minimum a WFS must be able to present features using GML.
4. The predicate or filter language will be defined in XML and be derived from CQL (Common Query Language) as described in the OpenGIS Catalogue Interface Implementation Specification.
5. The data store used to store geographic features should be opaque to client applications and their only view of the data should be through the WFS interface.
6. The subset of XPath expressions must be used for referencing the properties.

Search Service

A search service is for indexing and searching for GIS resources (e.g. data layer, a table, or a whole enterprise geodatabases). Web users can search for desired GIS resources by querying the search service, for example, by using keywords. A search service is different from a metadata catalog service. While both are for discovering GIS resources, a search service indexes the data, especially the attribute tables, directly, while a metadata catalog service indexes the metadata. (Open Geospatial Consortium Inc. 2007; Fu & Sun 2011.)

Image Service

An image service provides access to raster data, such as satellite imagery and digital elevation data, through a Web service. It supports raw data extraction and download, and often renders map images, as well. The image service is one of the most significant elements in the GIS server and without this service the GIS server would not be able to render the map images in various data formats requested by the client. (Fu & Sun 2011; Hazzard 2011.)

Geodata Synchronization Service

A geodata synchronization service exposes the ability to perform geodatabase replication operations, which is used to replicate or synchronize data updates between different geodatabases over the Internet. This kind of service is particularly useful in situations of distributing geodatabases in different locations - at different levels of governments, for instance. (Fu & Sun 2011; ESRI Support Center 2013.)

Analytical Services

Analytical geospatial Web services perform a variety of GIS functions, commonly used in conjunction geocoding, network analysis or geo-processing services. Geoprocessing services can accommodate any workflow and perform any tasks that the Web GIS application developer plans to use in the exploration (or geodesign) process. (DeMers 2009; Fu & Sun 2011; Hazzard 2011.)

Geocoding Service

Geocoding is a service of converting street addresses, postal codes or other administrative district regions to geographic coordinates, usually latitude and longitude, and therefore allowing to find and display addresses or area shapes on a map and to see how they relate to surrounding features. The geocoding service is not always a straightforward process and can be complicated by ambiguous addresses or addresses that exist in the database. So, geocoding results are usually a list of possible matches. A geocoding service is sometimes referred to as a locator service. The service may also support *reverse geocoding*, which is the process of finding the address pertaining to the geographic location. The word "Geocoding" is also used for the process of conversion to change analog spatial information into digital form. The methods in the process are map digitizing, scanning and field data collection. (Fu & Sun 2011; ESRI Support Center 2013.)

The digitizing process is capturing map data by tracing lines from a map by hand, resulting in a string of points with (x, y) coordinate values. *The scanning process* is capturing map data by scanning the image from a map by a scanning device. The result of scanning is a grid of pixels (raster image). Scanning involves placing a map on a glass plate while a light beam passes over it, measuring the reflected light intensity. Image size and resolution are important for scanning. *The field data collection method* is where map data are created by people, using GPS, Remote Sensing, Aerial Photography, and other field collection techniques. (Guan, 2006.)

Network Analysis Service

Network analysis is a service which helps to find the proper routes and facilities based on the information given to the service. The term “network” may refer to transportation network systems such as streets, and highways or any other geospatial information comprising networks of shapes. (ESRI, 2013.)

According to Fu & Sun (2011), a network analysis service can support the following functions:

- Network analysis – Function to find the proper routes and facilities based on the information provided.
- Routing analysis – Function to obtain directions for the shortest or fastest route between two or many places.

- Calculating analysis of network service areas - Function to calculate network service region that encompasses, for example, all accessible streets within a specified driving time.
- Finding the closest facility – Function to find the closest facilities to a position based on the driving time or distance.

Geometry and Geoprocessing Service

A geometry service is a service which performs geometric transformation and calculations such as buffering, simplifying, merging, splitting, calculating areas and lengths, and projecting coordinates to given geometry characteristics, for example, to calculate the flood elevations effects, using a geometry buffering, or measure the total acreage of some field owner. A geo-processing service provides the flexibility to share the functions or models to author locally. For example, it can be used to share all the Web service functions discussed earlier in this chapter. (DeMers, 2009; Fu & Sun, 2011; ESRI Support Center 2013.)

The OGC has defined Web Processing Service (WPS) specification, which is an interface standard, specifying the rules how a geospatial data is formulated (such as geometry data) in the geospatial processing services. The specification also defines how a Web client can request the executions of the methods, and how the outputs from the processes are handled. The specification also defines a standard interface that facilitates the publishing techniques of geospatial processes and client's discovery of and binding to those processes. The data required by the Web Processing Service can be delivered across a network or it can be available at the GIS server. (Schut, 2007.)

Metadata Catalogue Services

Metadata is data about GIS data, and it describes available data and services provided by the GIS server. A metadata catalogue service allows publishing and searching the GIS server metadata. The service enables sharing geospatial information and services provided by the GIS server. For example, a GIS server provider can publish metadata about map data and services for others to discover. (Open Geospatial Consortium Inc., 2007.)

2.6 Databases and Geospatial Data

The GIS database is the data storage that can hold a collection of geographic datasets of different types, such a basic vector data (points, lines and polygons) and raster data such as satellite and aerial images. Depending on the GIS database, it may support data types such as CAD, 3D, utility and transportation system data or GPS coordinates, and survey measurements. GIS databases range from small a single-user flat file systems (for example, Comma Separated Values, CSV file) to a distributed relational database management system (RDBMS) where data can be edited and accessed by multiple users simultaneously. While some GIS databases store only collections of individual features, others manage the data models that define the spatial relationships and behaviors that are critical to many GIS tasks and analytical operations. (Connolly & Begg 2005; Fu & Sun 2011.)

Fu & Sun (2011) have stated as follows:

> "The GIS database is the underlying support for WEB GIS applications. The answer delivered from a Web GIS application can only be as good as the quality of the information contained in the GIS database. While casual applications can rely on casual data sources, professional applications tend to need high-quality, authoritative, and up-to-date geographic information." - (Fu & Sun, 2011)

A GIS database generally includes the following capabilities (Fu & Sun, 2011):

- Stores a rich collection of spatial data in centralized or distributed systems
- Applies sophisticated rules and relationships to the data
- Defines geospatial relational models
- Maintains integrity of spatial data with a consistent and accurate database
- Works within a multi-user access and editing environment and supports versioning.
- Supports custom features and behaviors
- Provides a means for robust data security, backup, recovery, and rollback
- Maintains high performance when the volume data increases and the number of simultaneous users increases.

The spatial data are stored in databases usually as a combination of tabular data or geospatial data.

Tabular Data

Tabular data, also called attribute or descriptive data, is one of the most important elements in a GIS (Spatial Information Clearinghouse, 2004). The tabular data is statistical, numerical, or characteristic information that can be attributed to geospatial features. Similar to geospatial data the tabular data is stored by the GIS software to a RDBMS or a single file (for example, CSV) format. Depending on the application, attributes that may be useful to assign to a feature would be the population or temperature of an area (as illustrated in table 1), area measurements of a field, or types of network events in a particular area.

Table 1: Example of tabular data. Here the tabular (attribute) data columns are date, temperature and name of city.

Date	**Temperature**	**City**
30.4.2012	16,5	Helsinki
30.4.2012	17	Espoo
30.4.2012	17,5	Vantaa
30.4.2012	16	Korsoo

The GIS software allows the attribute data to be connected to the geospatial data and stored into the database in such a manner that it binds the attributes to a position (as presented in table 1, the City column stores the attribute value for the position). A GIS knows a particular location geographically from the storage of geospatial data (as in table 2). By linking the attribute data (table 1, the City columns data) to the geospatial data; the GIS can present the characteristics of a feature in the geospatial perspective.

More than one tabular database can be connected together when there is a common data field (for example, the name of city, or time). This allows the GIS to become a powerful spatial analysis tool. After integrating both, geospatial and tabular (or attribute) data, a GIS user has the capability to learn a great deal about the defined area of study.

Geospatial Data

Geographical information is different from other information in a way, that the data in geographical information (presenting object or phenomena) is referred to be a particular position in space. This position may be in some form of a coordinate system or pre-agreed name (as City name in table 1) which can be translated later into some system specific coordinates (for example, using table 2 values). Therefore, because of this aspect, the objects (or phenomena) can be visualized in some form of a graphical presentation (as on a map).

Table 2: Example of geographical information. The City names have Area Geometry information as a single location point in WGS84 latitude and longitude coordinate values.

City	**Area Geometry Information**
Helsinki	POINT(60.11,25.019)
Espoo	POINT(60.205145,24.656968)
Vantaa	POINT(60.319,24.969)

When an object's geospatial data is stored in a database, these geospatial data are usually divided into three aspects (see figure 8): spatial data, attribute data and temporal data.

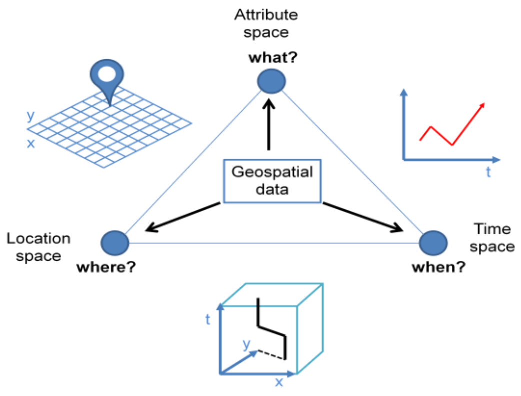


Figure 8: The geospatial data. The geospatial data consist of three aspects of data; location, attribute and time data. Modified from Kraak & Ormeling (2010)[120].

The spatial data refers to the object's geometrical features, which are location (as illustrated in table 2 where City position is given in the latitude and longitude coordinates) or location and dimension, or just a name of the place in the spatial space of the object. According to some predetermined criteria the dimension (or shape) of an object are usually presented as points, lines, polygons or volumes. The spatial data can answer questions, such as; *where is this happening* or *what is there*?

The attribute data refers to another, non geometrical characteristic belonging to the object or phenomena (see table 1, Temperature values of Cities). For example, the questions this attribute data can answer are; *what has changed since x*, *what is the temperature of y* or *what colour has this object?*

The temporal data (as in table 1, Date values) refers to some moment in time for which both the spatial and attribute data are valid, and thus it can answer questions, such as;

when has this happened, or *how long did this event take?* The importance of the temporality in geospatial data is stated by Kraak and Ormeling (2010) as follows:

> "All the geospatial data will be the subject to changes over time: the attribute information on an object can change over time (such as the composition of the population of an area), and even the object's location itself may change (for instance, the continental drift). The data's time stamp, is seen as the third major component, next to geometry and attribute values. Especially these days the interest in the data's temporal component increases because of the expanded number of time series available and the wish to analyse processes over time instead of during a single time slice."

Therefore, an object in a database with geospatial data can answer to the essential combination of questions: *Where [location]*, *What [attribute]*, and *When [time].* An object's location, attribute or time data can have various specific characteristics, such as different coordinate systems (as in table 2, Area Geometry values are in World Geodetic System 84 coordinate values), many variables and even different kinds of time. (Kraak & Ormeling, 2010.)

In addition to the essential questions; *Where*, *What* and *When*, one might also ask *Why* or *How?* However, answering these questions requires further analysis of the data and is the subject of geodesign as explained in chapter 2.

Geometry Data

As discussed in the beginning of this chapter, the GIS database is the data storage that can hold a collection of geographic datasets of different types. Therefore, the database system used in the GIS should support a way to handle the geometry data in an efficient way. For this geometry support, the OGC has defined the simple geometry operations and types what geospatially enabled database systems should support.

The geospatial data object in the database is a compound of three data type values (as discussed in the previous section); time, location, and attribute data. However, there are no strict or mandatory rules to have these all data types stored into the database to create a geospatial object. In a broader sense, the only required data to perform a geospatial object is to have attribute data which describes the object and links it to some place on the Earth. The OGC specification (Open Geospatial Consortium Inc., 2011) defines that this kind of geospatial data object is called as "feature" and it is an abstraction of real world phenomena. A characteristic of a feature is described that it has a name, a data type, and a value domain associated to it. A feature attribute for a feature instance also has an attribute value taken from the value domain. No restrictions are implied here as to the type of attributes a feature may have and, therefore, the “geometries” associated to feature are just one type of feature attribute.

Hence, for example, the tabular data listed in table 1, can be thought of a list of features, though with no geometries directly associated them. The hint of the place on the Earth is given in features attribute *"City"* values. Before table 1 feature values can be linked to place on the Earth (or some other place), table 2 geometry values must be used to set the table 1 values to the actual position in some system space (as a place on the Earth). In this manner, the example data in table 2 geometry data (in Area Geometry Information column) are spatial object data presenting the geometry set of feature, as the OGC defines (Open Geospatial Consortium Inc., 2011) it:

> "A geometric object consists of a geometric primitive, a collection of geometric primitives, or a geometric complex treated as a single entity. A geometric object may be the spatial representation of an object such as a feature or a significant part of a feature."

Therefore, geometry data in table 2 is presented in OGC’s compliant Well Known Text Representation (WKTR) format. In table 2, the City column’s data values are the geometry objects of points (POINT), which are the smallest unit to represent the geometry of the feature. The geometry object of point is a topological 0-dimensional geometric primitive, representing the object position on the predefined coordinate system.

The other geometry objects and their subtypes, as OCG has defined them, are depicted in figure 9. These objects are Curve, Surface, LineString, Polygon, PolyHedralSurface, MultiSurface, MultiPoint, Line, LinearRing, MultiPolygon and MultiLineString. Examples of these geometry types in WKTR format are presented in appendix 1.

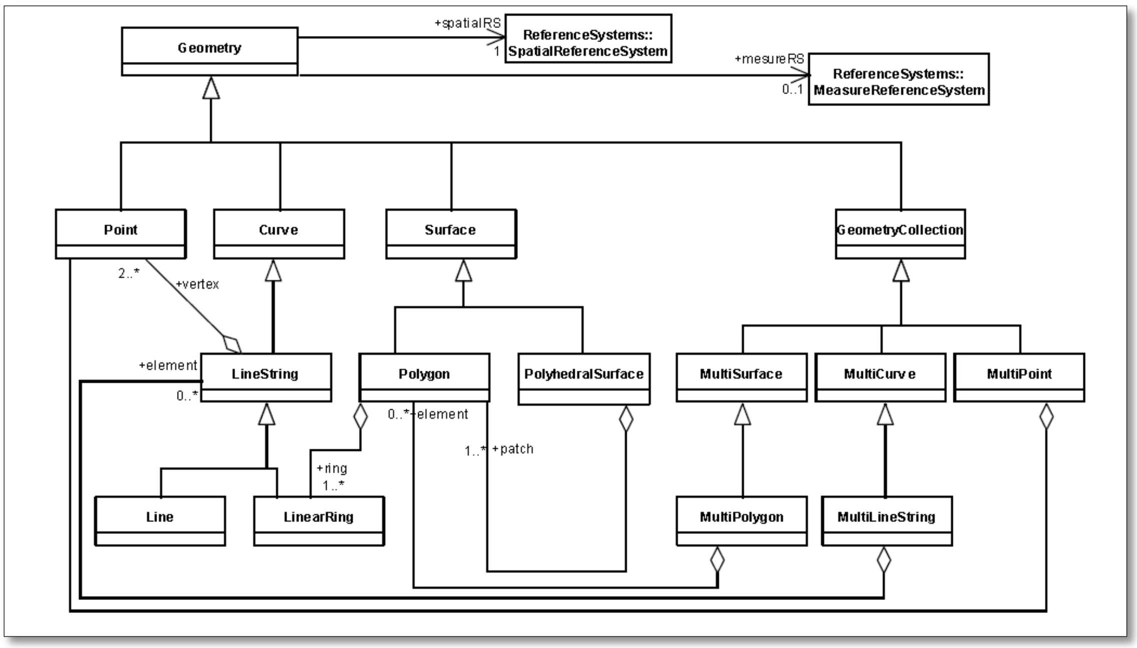


Figure 9: The SQL Geometry Type Hierarchy. Reprinted from OGC 06-10r4, p. 24

According to the OGC specification (OGC 06-10r4), all these geometry object types, gives a standardized way to describe the feature's geometry (a shape of form it presents) associated in some simple or multidimensional coordinate system:

> "Geometry is the root class of the hierarchy. Geometry is an abstract (non-instantiable) class. The instantiable subclasses of Geometry defined in this Standard are restricted to 0, 1 and 2-dimensional geometric objects that exist in 2, 3 or 4-dimensional coordinate space ($\Re_2$, $\Re_3$ or $\Re_4$). Geometry values in R_2 have points with coordinate values for x and y. Geometry values in R_3 have points with coordinate values for x, y and z or for x, y and m. Geometry values in R_4 have points with coordinate values for x, y, z and m. The interpretation of the coordinates is subject to the coordinate reference systems associated to the point. All coordinates within a geometry object should be in the same coordinate reference systems. Each coordinate shall be unambiguously associated to a coordinate reference system either directly or through its containing geometry. The z coordinate of a point is typically, but not necessarily, represents altitude or elevation. The m coordinate represents a measurement. All Geometry classes described in this standard are defined so that instances of Geometry are topologically closed,
>
> For example, all represented geometries include their boundary as point sets. This does not affect their representation, and open version of the same classes may be used in other circumstances, such as topological representations." (OGC 06-10r4, p. 24.)

The database supporting the OGC simple feature geometry definitions makes it very powerful for describing and storing graphical data associated with features or phenomena. The OGC specification also describes how the geometry data stored in the database must have Spatial Reference System Identifier (SRID) coupled in the data, which is the Spatial Reference System (SRS or CRS = Coordinate Reference System) number in integers, specifying the associated coordinate reference system used in the coordinate values in the geometry data type presentation.

The spatialreference.org Website maintains and lists over a thousand different coordinate systems used around the globe within various map projections (Spatial Reference Org., 2013). While most of the spatial reference systems are based on some map coordinate system, they can also be user generated coordination systems such as Full-HD screen resolution (1920x1080 pixels) coordinates where x are values between 0 and 1919, and y values are between 0 and 1079.

In order for the GIS enabled database to be able to process or convert geometry coordinates, it must support the projection systems. This is done by maintaining the SPATIAL_REF_SYS metadata table (or similar) in database. In general most of the standard projection coordinate systems are preinstalled in the databases supporting geospatial information and users of the database may use the SQL geometry functions in a very straight-forward way without the need to know the existence of SRS. (OSGeo, 2013)

Including the geometry data type support and coordinate projection systems, the GIS-enabled database shall also support the primary geometric routine methods specified by the OGC. These geometrical methods are given tools to play with the geometry data, for example, to test if two different feature geometries intersect, touch, cross, or overlap each other or if they are part of some other geometry unions.

Geospatially Featured Databases

According to the Technology Survey of GIS (TSG) made by Tomi Salmi (2012), the database is the central point of storing data nowadays. Yet geospatially featured databases have not been available until after 2005. Today, almost every mainstream database supports geospatial data management features and most of GIS servers available in the markets also support them. (Salmi, 2012.)

Salmi (2012) lists nine database brands supporting geospatial features in some level, as show in table 3.

Table 3: Geospatial featured databases. Data gathered from Salmi (2012)

Developer	Brand	Type	License	More Information	Notes	
D. Richard Hipp	SQLite	RDBMS	Public domain	www.sqlite.org	*Database for mobile device. Very compact sofwate package (~300kB)*	
ESRI	ArcSDE		Proprietary	www.esri.com/software/arcgis/arcsde	*ArcSDE (Spatial Database Engine) technology is a core component of ArcGIS for Server. It manages spatial data in a relational database management system (RDBMS)*	*Supported Databases: DB2 Informix Oracle PostgreSQL SQL Server and SQL Server Express*
IBM	DB2	RDBMS	Proprietary	www-01ibm.com/software/data/spatial/db2spatial/		
IBM	Informix Spatial	RDBMS	Proprietary	www-01.ibm.com/software/data/informix/spatial/		
Microsoft	SQL Server 2008 R2	RDBMS	Proprietary; both commercial and freeware editions are available	www.microsoft.com/en-us/sqlserver/	*OGC compliant*	*good geospatial features*
Open Source project led by OSGeo	PostGIS	ORDBMS	GNU	postgis.refractions.net/	*OGC compliant Add On for PostgreSQL*	*Free and very good geospatial features*
Oracle	MySQL	RDBMS	GPL or Licence for OEMs, ISVs and VARs	www.mysql.com		*Geospatial features are very limited*
Oracle	Locator	RDBMS	Proprietary	www.oracle.com	*A feature of Oracle Database 11g Standard Edition*	*Limited geospatial functions*
Oracle	Spatial	RDBMS	Proprietary	www.oracle.com	*A pay add-on feature of Oracle Database 11g Standard Edition. Adds complex spatial analysis , raster file storing, routing and geocoding support*	*Most expensive, but maybe best geospatial database on market*

According to the TSG (2011) report, Oracle Database 11g Standard Edition with Oracle Spatial add-on is most advanced geospatial database and is supported by every GIS server found in the market (including the Open Source systems). However, Oracle is very expensive and intended for very heavy use and massive data manipulation.

ESRI’s ArcSDE is a database engine providing a spatial data interface for multiple database brands, including DB2 and Informix. ArcSDE engine is provided as part of the ESRI ArcGIS software package.

Every Microsoft SQL Server edition, starting from release 2008 (r2), supports many of the OGC database specifications, as well. From the pricing point of view, it is a very good choice to use in business solutions. Microsoft offers various license model options depending on the SQL Server edition acquired. The one significant difference with licensing strategy compared to the Oracle model is that Microsoft does not restrict the virtual server usage for single processors as Oracle does (licence fee per used processor). (Salmi, 2012.)

Considering the Open Source database systems, the PostgreSQL with PostGIS add-on is free, and supports widest the OGC specifications for geodatabases. The MySQL database server software is also available for free under the GPL license; however its geospatial features and decentralization abilities are very limited, for example, compared to PostgreSQL.

The SQLite database is claimed to be most widely deployed SQL database engine in the world (SQLite, 2013) and it is distributed under the Public Domain license. The SQLite also supports some geospatial features, such as storing geometrical type data. The advantage of SQLite database for others is that it is a very compact software library (under 300kB) that implements a self-contained, zero-configuration and transactional database engine. The compact size makes it very useful and practical to use in mobile device applications, which has a very limited amount of memory available.

2.7 GIS Packages and Clients

The client in a GIS application package can play two roles. First, it represents the end-user interface for the entire system. It interacts with the user inputs, sends requests to the server, and presents the results to the user. Most end users do not know, and do not need to know; the particulars of the back-end server(s). All that they know about the system is how fast, stable, and user-friendly the client is. Second, the client can also perform some geospatial processing tasks, such as dynamic classification for thematic mapping, cluster, and heat-map analysis. GIS clients are typically Web browsers but can also be desktop applications, mobile applications, or even server applications (when a server acts as a client of another server). (DeMers 2009; Fu & Sun 2011.)

GIS application packages usually provide tools for users that perform processes beyond mapping. These applications range from the common types, such as finding address or place-name (geocoding), routing, and searching point of interest that match certain criteria, to other advanced tools that implement specific business logic and data exploration for an enterprise. The chosen solution varies as to whether the methods should be performed by the server or the browser.

Fu & Sun (2011) divides the client applications into two forms, Thin Clients and Thick Clients. The differences between these two are that the Thin Client relies more on the server to perform most of the work by leaving the client to do the least amount of processing, and the Thick Client relies more on the client, rather than the server, to perform most functions.

In the Thin Client architecture, the client simply sends the user's request to the server. The server does the processing, such as generating a map and performing analysis. The results, typically in HTML format embedded with GIF, PNG or JPEG images, are then returned to the client and displayed for the user. Therefore, Thin Client Application architecture is dependent on GIS servers and the services it provides and on the GIS databases it can use. In the Thick Client architecture, the client requests the source data (for example, the coordinates of vector data) directly from the database or flat files, and then renders maps and performs analysis on the client side (Fu & Sun, 2011). Thin Client is a more independent solution and does not necessarily require any communication to GIS servers.

As the Web technology has advanced rapidly, the client-side technology, such as plug-ins and JavaScript, has become more powerful and is, therefore, able to handle greater workloads. Thereby, a contemporary general design strategy has been breaking down the workload into several categories and properly distributing it between the server and client as illustrated in figure 10.

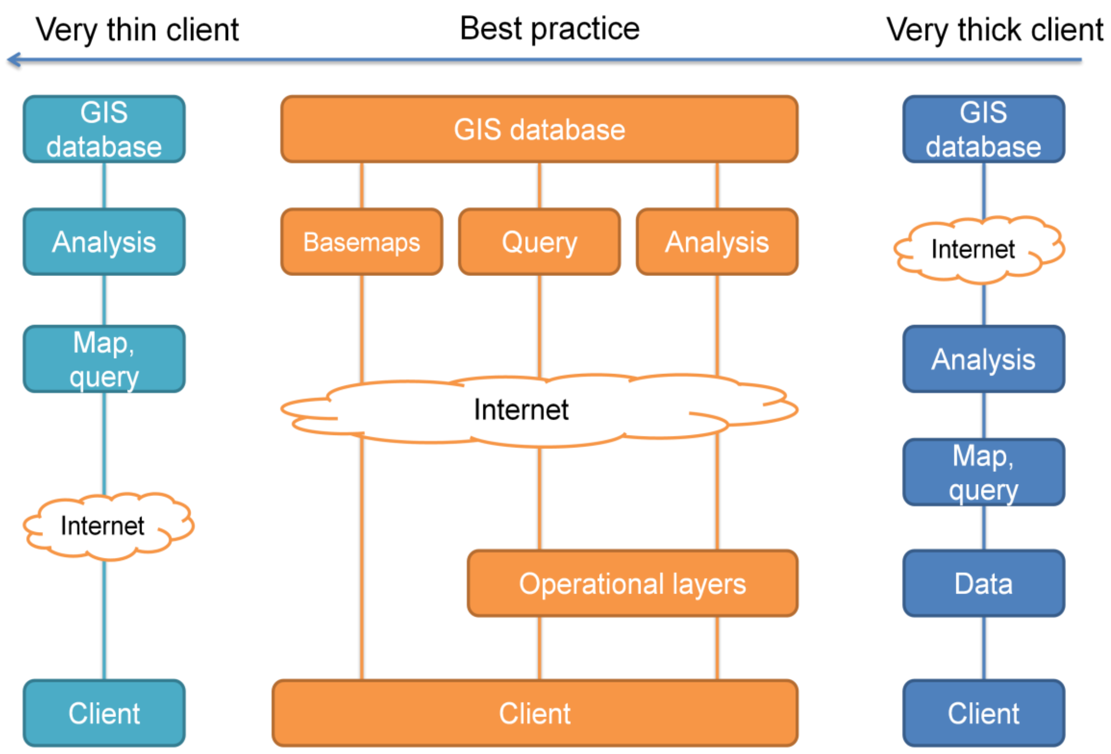


Figure 10: Client architectures. In the extreme thin client architecture, most GIS functions are performed by the server, while the opposite is true for extreme thick client architecture. Modified from Fu & Sun (2011)[41]

The current best practice recommends that base maps are handled by the GIS server and operational layers are typically rendered by the browsers, unless the data size is too large for browsers to handle. Essentially easier functions are handled by the browsers and complex functions are handled by the GIS server. (Fu & Sun, 2011.)

It is widely acknowledged that geospatial technology can greatly benefit non geospatial industries once the GIS is easily and seamlessly integrated into other information systems, but that potential has not yet been fully reached. The complexity of having to copy GIS data locally and to install GIS software components locally remains a barrier in many situations to using the GIS.

Geospatial Web services running on GIS servers, on the other hand, hide the complexity of GIS data and functionality, leaving it to be handled locally or remotely on other servers, while exposing a Web programming interface for easy integration. This means that other IT systems can simply access mapping, data, and geo-processing Web services from a variety of sources without having to deal locally with the geospatial complexity. Web services allow the GIS and its spatial processing to be integrated flexibly and extensively with IT business systems, such as ERP (Enterprise Resource

Planning) and CRM (Customer Relationship Management), serving as the best building blocks for SOA (Service Object Architecture). This openness and flexibility can greatly expand the GIS market. (Fu & Sun, 2011.)

GIS Applications

The purpose of on GIS application is to provide the tools for users to perform processes which include geospatial data. As discussed in previous section, these processes range from the common types, such as finding addresses, place-names or routes, or search for point of interest that matches certain criteria or even plan and create maps for land usage. All these processes need a variable list of functions that the GIS application needs to support to perform the required process. For example, Fu & Sun (2011) lists four different common functions to do with on GIS application: mapping and query, collecting geospatial information, disseminating geospatial information and geospatial analysis.

The mapping and query functions are the most commonly used ones because GIS data and analysis results are usually presented as maps. Mapping functions hold a wide collection of use cases. For example, at one edge there are cartographers creating the maps using GIS applications to visualize geographical data. In the other edge, there are map users to plan routes or area usages based on the visualized maps created by cartographers. In these examples, the requirements of GIS application properties are different. A professional cartographer's or data analyst's requirements for tools to manipulate map data are very different from a map user casually just planning a route between destinations or finding an interesting place to visit on the following summer holiday.

Collecting geospatial information is the process where the GIS application provides methods and functions to create and assemble geospatial data to the database. For example, applications like this are Wikimapia (Wikimapia, 2013), OpenStreetMap (OpenStreetMap Org., 2013), Twitter (Twitter Inc., 2013) and Flickr (Yahoo Inc., 2013). Wikimapia and OpenStreetMap are open source Web applications for Internet users to collect and use Web maps freely as an option for Google, Yahoo and Microsoft. Twitter is a message-sharing service and Flickr a photo-sharing service which both may include geospatial information about the places in where the event occurred.

Collecting geospatial information can be a very complex, and time-consuming process where the data collection starts from measuring the geographical data in the field, using field tools by field engineers such as laser meters, GPS sensors or aerial imagery, and ends to store the collected data in an appropriate form to the database using the aid of the GIS application. Therefore, the capability of a GIS application to support collections of information through databases or Web API sources is essential, because, without any information, the GIS application is useless. (Spatial Information Clearinghouse 2004;DeMers 2009; Fu & Sun, 2011; Caitlin Demsey 2012.)

The dissemination of geospatial information process is part of the publishing process and dependent on the information sharing policy selected. The dissemination is done using the result of GIS tasks made by a user. The dissemination process can be as simple as just saving the GIS application result to the database or printing it on the paper. However the real dissemination applies to a GIS application or systems where the processed data is shared for use for other users or systems. According to Fu & Sun (2011), Web GIS is an ideal platform for wide distribution of information.

Examples of dissemination of geospatial information are world and street maps by Google, national terrain maps and other geographical information shared by National Land Survey of Finland or weather information provided by Finnish Meteorological Institute. All information collected in the above examples are results of a huge amount of work, but they are shared for public use for others via Web service APIs or Web clients. The reasons for opening and disseminating the information as open (to the Internet) or closed data (dissemination for a limited group of users) depends on the data and the goal of the result made by the geospatial analysis. Whatever the decision of the dissemination policy is, it would be a waste of money and time if no dissemination was done at all or just for a very small, limited number of people. (Coppock & Rhind, 1991; Miller 2012.)

Geospatial analysis is a process that provides analytical functions such as measuring distances and areas, finding the optimum driving path as in navigation, finding the location of an address or place, or count buffers of geographical features. From a GIS application point of view, these functions are part of the application software or separated API functions handled by the GIS server on the client call (Thin client vs. Thick client). For example, GPS (Global Positioning System) based navigation devices are mixture of thin client and thick client (in sense of GIS application architecture, as illustrated in figure 8). They are devices which hold on embedded database containing the map data with locations, places, streets and other geoinformation, and built-in software to visualize the driving and navigation. The goals of geospatial analysis they perform are to find the fastest or shortest routes between given destinations by users, to show the current position, speed and time estimation left for the destination, on the map, based on the GPS information available. Embedded GPS navigation devices are infrequently Web-enabled, and thus they lag the dissemination functions. However, the GPS navigation software in Smartphone may support the dissemination of user routes, location or other information.

The selection of GIS application depends on the requirements of the GIS processes and functions needed by users as discussed in this chapter. The needs of a cartographer, field engineer, data analysts or simple home user are different from the GIS application to perform the goal they want. Therefore, making a decision as to what kind GIS application or mixes of applications are required, depend on the many factors.

Table 4 lists some well-known GIS applications on the market, and as the list exemplifies, there are several of differences between GIS applications. Examples of the thin client architectures are the Google Map, Yahoo Map, Microsoft Bing and OpenStreetMap Web applications for home users and Web developers. For home users, they provide a simple Web user interface to do queries from places, locations and navigation guides, pinpoint of interests and share them with other Web users over the Internet. For developers, they are easy-access mapping and geocoding sources. What they do well, is providing maps, streets, cities generalized from around the world. However, they do not provide all information of paths, forests and fields that one may need in accurate navigation. These cloud services offer a very limited number of geo analysis functions, and thus are as provided. All extra-geo manipulation methods and functions must be programmed in the client side. They also lag the privacy in the sense of trust. Users and companies worrying about sharing their privacy or other sensitive data would not use these applications to distribute their data by any means. There are also a number of restrictions as to how these services can be used. Google, for example, states in their purchase agreement that a customer must not use the service for high risk activities or use the service or any content in any customer implementations for real-time navigation (Google Inc., 2013).

Table 4: Some well-known GIS applications.

GIS Application	Usage Type	Software Nature	Licensing	Notes	Source
Autodesk	Desktop	Native Applications for Windows (dependant of Client, many others operating systems may be supported as well, such as Android)	Commercial	Enterprice solution package, including GIS server and client applications.	http://usa.autodesk.com/gis-design-server/
ESRI ArcDesktop	Versatile		Commercial	ESRI provides complete sofware system, including GIS server, client applications including Web, Mobile and Desktop. Integrates for Microsoft office tools	www.esri.com/products
Google Map	Web Client	Web Application	AddWare / Commercial	Cloud service for Internet users and developers	maps.google.fi
MapInfo	Desktop	Microsoft Windows Application	Commercial	Windows–based mapping and geographic analysis application. Designed to easily visualise the relationships between data and geography,	www.pbinsight.com/welcome/mapinfo/
OpenJUMP	Desktop	Java Application	OpenSource	Simpe Java application for Mapping and geoanalysis	www.openjump.org
OpenStreetMap	Web Client	Web Application	OpenSource	Free cloud service for Internet users and developers. OpenStreetMap is project that creates and distributes free geographic data for the world.	http://www.openstreetmap.org/
Quamtum GIS	Desktop	User Interface based on Qt Toolkit	OpenSource	Versatile software package, including GIS server and client applications, and many analyzing plugins. Supports OGC specifications and many ESRI data sources. Contantly new updates and plugins	www.qgis.org
uDIG	Desktop	Eclibse Plugin	OpenSource	The goal of uDig is to provide a complete Java solution for desktop GIS data access, editing, and viewing.	udig.refractions.net
Yahoo Map	Web Client	Web Application	AddWare / Commercial	Cloud service for Internet users and developers	maps.yahoo.com

In the opposite of the thin client architecture applications are MapInfo, uDIG and Open-Jump. They are meant for users who manipulate maps or analyses geospatial-enabled data. The disadvantages on these applications are that they do not integrate well to other IT systems nor support dissemination through Web services. The advantages on these applications are that they are good for viewing and visualizing the relations between data and geography.

In the middle (as illustrated in figure 10.) are GIS applications (or more strictly speaking GIS packages) such as ESRI, Quantum GIS and Autodesk which are meant for enterprise use and companies doing a number of geospatial work. The GIS applications included in these product families are solutions for GIS servers, databases and user client applications from native desktop clients to Web clients. From apart of two others, the ESRI also provides a number of geographical data such as maps, locations, street, places etc. ESRI products integrate well into Microsoft products, for example, PowerPoint or Excel plug-ins, to visualize data on maps.

The advantages of GIS applications are that only imagination is the limit as to what one can do with these products. They support well for any GIS task required by users. However, the disadvantages are the pricing (except the Quantum GIS, which is Open Source) and the learning curve required to use these applications with all the power they provide.

To summarize, there are a number of GIS applications available in the market; some are free, and some cost a huge amount of money. Some are meant just for visualising the data, and some are meant to create the data. Some encompass more possibilities than others and some need more learning than others. Some include all the GIS software components and some just a few of them. Therefore, making the decision about what kind GIS application package is required may be a very hard task to perform.

Hans Bestebreurtje states in his Master's Thesis covering the GIS Project Management (1997):

> "Choosing the appropriate development methodology depends on the situation. The "one fits all" methodology unfortunately does not exist. Currently, combinations of information engineering and proto typing are used. The basic thought behind information engineering is that data is the most stable factor when developing an information system. This method specially is useful in a project with the following characteristics:
>
> - High uncertainty of specifications;
> - Need of decision support systems;
> - Low expertise in this field of current users;
> - High level of uncertainty concerning the exact specifications of GIS." (Bestebreurtje, 1997)

The statement applies as well to the problematic case of choosing the right GIS application packages.

To find an answer to what kind of GIS application package or application architecture should be chosen, one should first answer the following questions (Bestebreurtje 1997; DeMers 2009; Fu & Sun, 2011):

- What is the value-add for business to do the geospatial analysis versus of the cost of the selected GIS application to do the job well,
- What data and in which form is the data provided and is it sensitive,
- What GIS functions are required to do the analysis,
- How are the end results of the analysis used by others,
- What kind of needs are there for dissemination,

- What kind integration is required for the existing IT systems is, and
- Who participate in the GIS project and what are their roles in it?

3 Fundamentals of the GIS Application

3.1 Basemaps and Operational Layers

The questions the GIS are designed to answer remain fundamentally geographic ones. Therefore, the goals of any geographic information system are to help users to input, present and analyze the georeferenced data. According to DeMers (2009), GIS helps users to think spatially and become more sensitive for the geospatial locations, patterns and distributions of the data. These geospatial data are visualized using maps, which is a compact and elegant method of communicating geospatial information. Equally, Kraak and Ormeling (2010) claim that is not impossible to get an overview of an area or place on the Earth without a map:

> "It is not possible to get an overview of an area in any way other than by consulting a map. A map places geospatial data, i.e. data about objects or phenomena of which one knows their location on the Earth, in their correct relationship to one another." (Kraak & Ormeling, 2010, p. 38)

Kraak & Ormeling (2010) also outline that geospatial analysis often begins with maps, because maps support judging intermediate analysis results, as well as present the final results. In other words, maps play a major role in the process of geospatial analysis. Maps are a direct and interactive interface to GISs, a sort of graphical user interface (GUI) with a geospatial dimension.

In computing, a graphical user interface is defined to be a type of user interface that allows users to interact with computing systems using images and graphics rather than text commands, thus making it an integral part of the underlying system (Wikipedia, 2013). In a GIS perspective, these images and graphics are digitised maps visualising the geospatial data in different forms. The form of maps, data visualising capabilities and available functions to play with data depend on the GIS client application and the GUI features enabled on it to gain the purpose the application is meant for. Despite the different features and functions between many GIS client applications the fundamentals in GUIs are same; they provide a means to present data on separate graphical layers, which are visually placed on top of each other. The first layer is called *basemap* (or *baselayer*), and is usually loaded with the map image data, providing the locational coordinate reference system for the map context to be presented. (DeMers 2009; Fu & Sun 2011; Hazzard 2011.)

Basemaps serve as the foundation for all subsequent operations usually done for the layers on top of it. These other overlaying layers are called *Operational* (or *Overlay*) layers and are drawn on top of the base map containing the themes that the end user will view and work with.

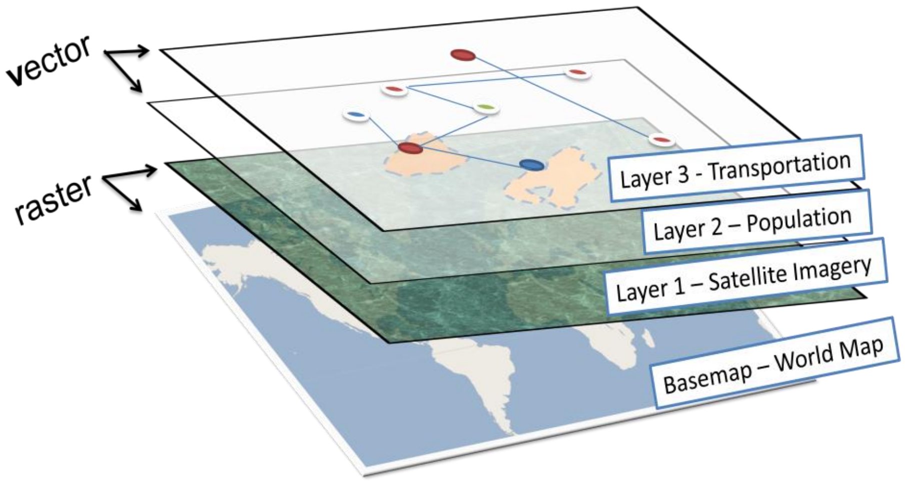


Figure 11: Fundamentals of the GIS application's graphical user interface. Fundamentals on any GIS client application's graphical user interface are same; they provide means to present data on separate graphical layers, which are visually placed on top of each other. The first layer is usually called for *basemap* or *baselayer* and is usually loaded with the map image data, providing the locational coordinate reference system for the map context to be presented. All the layers may contain raster or vector images.

All the layers may present raster or vector images as illustrated in figure 11, where the Basemap and Layer1 data contains raster images (scanned image of World Map and Satellite Imagery) and Layer2 and Layer3 data presents vector images (Population as area polygons and Transportation network as lines and points). The differences between raster and vector images are discussed in more depth in section 3.3.

Fu & Sun (2011), list the following type of operational layers as follows:

- **Data of observations or sensor feeds layers** are layers that show information that reflects the status or situational awareness. For example, crime location, traffic sensor feeds, real-time weather, observations from equipment or made by workers in the field, inspection results, addresses of customers, disease locations, air quality and pollution monitors, twitter feeds, GPS data, mobile or fixed network operational status and etc. These layers are the data layers that end users use for analysis or exploration to find phenomena or as base information to do geodesign.

- **Editing layers** are layers that users work with, for example, to edit features.

- **Query result layers** are layers showing a returned set of records from the server to respond to application queries. These records can include a set of individual features or attribute records. Users often display and work with these results as map graphics in their GIS applications.

- **Result layers** are layers derived from the analytical models. GIS analysis can be performed to derive new information that can be added as a new map layer and then explored, visualized, interpreted, and compared by end users. The result layers are used for dissemination to share the information discovered.

3.2 Map Extent

Typically, layer data are small in size or are only displayed when users zoom to a certain level. The visible area of layers presented for the user is called *map extent*. The limits of the map extent are defined in the coordinate system of the map used in basemap layer. The map extent is a rectangular shape defining a maximum and minimum width and height of the visible portion of the layers data presented to the user (as illustrated in figure 12). Technically, the map extent is also used to control the amount of map data sent over the network between clients and servers. (Kraak & Ormeling 2010; Hazzard 2011; ESRI 2013.)

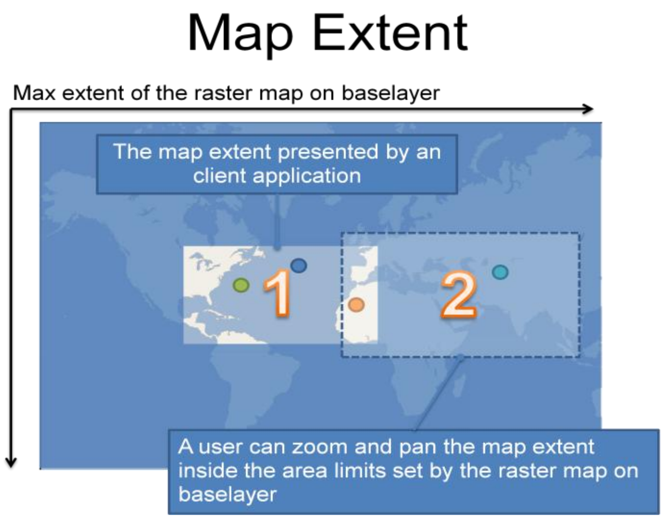

Map Extent
Max extent of the raster map on baselayer
The map extent presented by an client application
1
2
A user can zoom and pan the map extent inside the area limits set by the raster map on baselayer

Figure 12: Map extent. Map extent is technically constraining the data presented in graphical user interface and the information sent between a GIS client and server in Web GIS. When the user changes map extent size and position, the client will discover the data that belongs to this new area, and renders it to the user's view.

In Web GIS, the data is streamed to the GIS client application from the GIS server and then rendered and managed by the GIS client application. For example, in figure 10, the basemap is presenting the image of the world map. However only the area defined by map extent parameters are rendered to show in the GUI, limiting also the map data presented in operational layers overlaid on the basemap (green, blue and yellow dots). The user can zoom and pan the map extent in the limits of map extents set by a basemap. Every time the parameters of the map extent changes, the client application resolves what map data belongs to this new map extent and re-renders it. The map extends size and view location may also change automatically by the client in the result of some analysis made by the user. (Hazzard, 2011.)

There are many reasons for use the map extent to control the data exchange between a GIS client application and server. First, operational layers are often dynamic, and thus generally should not be cached in advance as the opposite of basemaps presenting maps (which are usually static raster images). Second, users need to interact with these layers to conduct their daily work. They expect operational layers to be responsive, reporting details on a mouse click and showing links to certain functions. The dynamic rendering and user interaction fall on the client side and can be implemented using browser APIs such as ArcGIS API or OpenLayers library for JavaScript. Occasionally layer data can be too large to be effectively rendered by the browser. In this case, the layer data should be rendered by the assistant GIS server using a Web service such as WMS. (Fu & Sun, 2011.)

3.3 Vector and Raster Images

As described in section 3.1, the GIS client application may present the graphical data in the data layer in raster or vector image form. Differences between raster and vector images are that raster images are constructed using a grid of pixels (or dots on paper) to define the image (as illustrated in figure 13) and vector images are constructed using mathematical equations to define the shapes and points of the image.

In the raster image, each pixel is assigned a color value and all of the pixels together form the image. The raster image resolution depends on the number of pixels used per image.

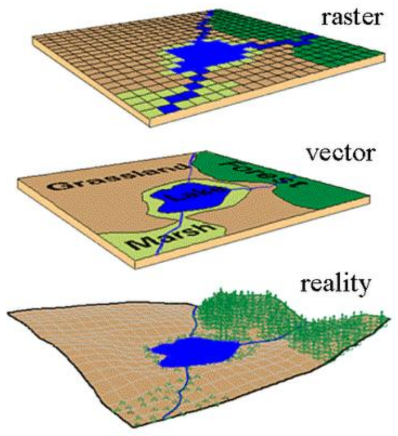


Figure 13: Vector image vs. raster image. Reprinted from Kevin (2010)

In the GIS terminology, the pixel is also referred to as *grid cell* representing a real geographic space (DeMers, 2009). The raster image file size is dependent on the number of the pixels and color depth used in pixels.

The raster image shape is rectangular, and each pixel in it can be referenced using planar x, y coordinate system as illustrated in Figure 14. The pixel color in each grid cell can be presented, for example, using the RGB (Red, Green, and Blue) color values. The intensity value of each color component may vary between 0 and 255 (if the chosen depth of color system is presented in 8-bits). Therefore, pixel at coordinates x=2, y=1 (see figure 14.) would have the grid cell values 0, 0, 0, presenting the pixel with the black color, and all the pixels in the first row y=0, x=0 to 5, would have grid cell values 255,255,255, (or FF, FF, FF in hexadecimal form) presenting the white color. (Watkins 2004-2010; Cohen 2009).

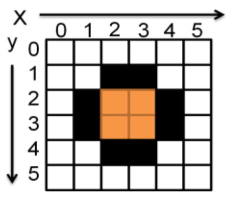


Figure 14: Raster image. In raster image, the pixel color in each grid cell can be presented using the RGB (Red, Green, and Blue) color values.

In the vector format, the image in figure 14 can be presented as a circle, where the center point coordinate values would be near x=2 and y=2, with a radius of 2 pixels.

DeMers (2009) has weighted the benefits between raster and vector data as follows:

- Raster data takes up more computer space than vector data,
- Raster is faster (computationally) than vector,
- Raster is compatible with much scanned and remotely sensed data,
- Raster is less spatially accurate than vector,
- Raster date give more options to work with surfaces than vector,
- Raster generally provides visually less desirable output (especially with its coarser resolution) than vector,
- Raster is more powerful for modeling than vector,
- Raster is more compatible with the printer output technology and less compatible with the plotter output technology than vector.

Georeferencing the Image

Georeferencing is the process of scaling, rotating, translating and deskewing the image to match a particular size and position. The word is used to describe the process of *referencing* a map image to a *geo*graphic location. Georeferencing is essential in the GIS, since all information must be linked to the Earth's surface. For example, all raster images used in the GIS must be georeferenced before the GIS can do any spatial analysis for it. (Muice, 2010)

The Georeferencing processes for raster images are usually done by using an appropriate GIS client application, which supports georeferencing. In this process, the user selects a raster image (for example aerial imagery taken from a helicopter using a digital camera) and assigns the real-world coordinates to each (or some) pixel of the raster. Usually these coordinates are obtained from field surveys (such as collection of known places on the picture and their GPS coordinates). In some cases, digitally scanned maps can be used, as well. In these cases, the user may obtain the coordinates from the markings on the map image itself. Using these coordinates or Ground Control Points (GCP), the raster image is warped and made to fit within the chosen coordinate system. (Gandhi, 2013)

After the process, the raster image must be stored in image format (sometimes called as GIS file formats) which supports this new supplementary geodata included into it, such as GeoTIFF, ESRI World File, JPEG2000 (OpenPlans Org., 2013). The complete list of all GIS supported raster image formats are listed in OpenPlans Org.(2013).

3.4 Visualizing Geospatial Data

3.4.1 Maps and Data Visualization

Visualization is defined to be as the communication of information using graphical representations (Kraak & Ormeling, 2010; Yau, 2011). A single picture can contain a wealth of information, as commonly phrased; "*One picture is worth a thousand words*" (Martin, 2013). A single picture can be also processed much more quickly than a comparable page of words because the image interpretation is performed in parallel within the human perceptual system.

Yau (2011) has defined that visualization is one of the best ways for people to explore and try to understand a large dataset because human brains are very good at finding patterns in a visual space. It can be a cross-border tool for exchange information, being independent of the local language, which may be understood by people with no common language (Ward, et al, 2010). Maps (a form of picture) are compact and elegant methods to visualize geographical information. If a map is designed correctly, it can tell a more complete story, and answer the preceding questions, just by a glance, than formal statistical methods (DeMers, 2009). Under the circumstances, what comes to the GIS and maps, the ultimate role of any GIS is in helping technically the users to produce maps and data visualizations of data containing geospatial information.

3.4.2 Map Types

Friendly (2009) has discussed that data visualization is the science of visual representation of “data”, defined as information which has been abstracted in some schematic form, including attributes or variables for the units of information. One of the subsections of data visualization types is maps, primarily concerned with representation constrained to a spatial domain and its goals to the visual representation for exploration and discovery. Map types range from the simple mapping of locations (land mass, rivers, terrain) to spatial distributions of geographic characteristics (species, disease, ecosystems), to the wide variety of graphic methods used to portray patterns, trends, and indications. (Friendly 2009; Skau 2012).

To answer the question how and what kind maps are used in data visualization, at full length, would be worth of its own master’s thesis, and thus it is not discussed in more depth in this thesis. However, a few of the common types of maps used in data visualization are discussed superficially to understand the goal of the GIS, which is data visualization through the maps.

Skau (2012) has discussed six map types commonly used to visualize the data (figure 15); Colorpleth, Cartogram, Connection, Pinpoint, Metro and Isopleth.

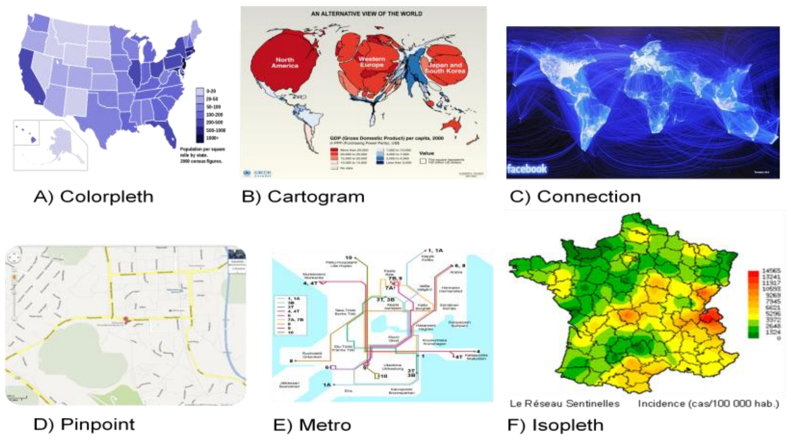


Figure 15: A) Colorpleth map, B) Cartogram map, C) Connection map, D) Pinpoint map, E) Metro map. and F) Isolate map. Reprinted from Skau (2012), Tikunov, (2002), Butler (2010), Google Maps (2013), Wikipedia Org. (2013) and FEM Wiki (2013).

The choropleth maps (Greek: choro = area, pleth = value) are one of the most frequently used maps in infographic style visualizations (Skau, 2012). A color scale is assigned to categorical or numerical data, and the value for each region are used to color the region as illustrated in figure 15, A. These maps usually use political boundaries as the regions. *The cartogram map* (figure 15, B) is a subtype of colorpleth maps. Ward, et al. (2010) has defined that cartograms are a specific type of map transformation, where the regions are resized according to a geographically related input variables (Ward, et al. 2010; Skau 2012). *The pinpoint map* (figure 15, D) is a map type that shows the exact location of things. These map types are popular to show the exact location of the items (Skau 2012; Google Maps 2013). *The connection map* (figure 15, C) is similar as pinpoint map, although the points have connections between pinpoints. In many examples of this technique, the connections are abstract presentations of phenomena such as Facebook connection or tweet replies where the connections are represented with an arc or straight line (Skau, 2012). *The metro maps* (figure 15, E) are another version of connection maps. The connections are most important in these maps, although the precise station location is not. Metro and transit maps are always manually created by designers who simplify the routes down to lines at a few different angles (Skau, 2012). *The isopleth map* (figure 15, F, is also called to ‘heat’ or ‘density’ map) is the map type that shows a range of quantity. Usually this map type is used for to illustrate weather data (Rouse 2011; Skau 2012).

3.4.3 Map Projections

Visualizing geospatial data, map projections play a critical role. Map projections are concerned with mapping the positions on the globe to positions on the screen (flat surface). Ward, et al. (2010) have defined that a map projection is

$$\boldsymbol{\Pi} : (\boldsymbol{\lambda}, \boldsymbol{\varphi}) \rightarrow (\boldsymbol{x}, \boldsymbol{y})$$

Where ($\boldsymbol{\lambda}$) is fixed to the interval [-180, 180] of degrees of longitude in radians, negative values standing for western degrees and positive values for eastern degrees, ($\boldsymbol{\varphi}$) is fixed to the interval [-90, 90] of degrees of latitude in radians, negative values standing for southern degrees and positive values for northern degrees, ($\boldsymbol{x}$) is a horizontal axis of the two-dimensional, and ($\boldsymbol{y}$) is a vertical axis of the two-dimensional map.

The term *latitude* is defined to be the angular distance, in degrees, minutes, and seconds of a point north or south of the Equator. Lines of latitude are often referred to as parallels as illustrated in figure 16. The term longitude is defined to be the angular distance, in degrees, minutes, and seconds, of a point east or west of the Prime (*Greenwich*) Meridian. Lines of longitude are often referred to as meridians. (Clarke 2005; Ward, et al. 2010.)

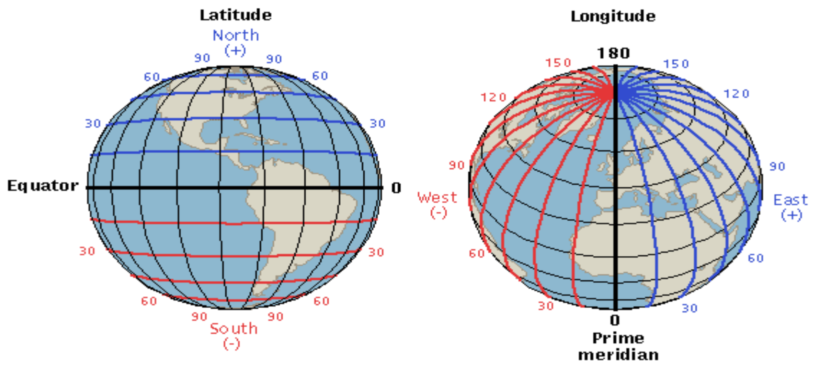


Figure 16: Definitions of Latitude and Longitude. Latitude lines are imaginary lines parallel to the equator divided into 90 degrees above and below the equator. Longitude lines are perpendicular to the lines of latitude. All lines of longitudes intersect at the North Pole and South Pole. Longitude is divided into 360 degrees (-180 to 180 degrees) starting from the Prime Meridian (0 meridian). Reprinted from Clarke (2005)[25]

Map projections may also have different properties and type of surfaces onto which the sphere is projected (Ward, et al., 2010). According to Hazard (2011), the most common surface types used are a cylinder, cone or plane (*azimuthal*) as illustrated in figure 17.

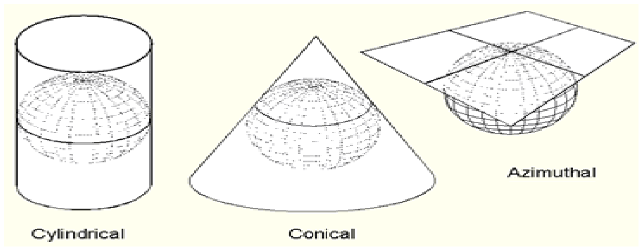


Figure 17: Three of the most common types of projection used in projecting maps. Reprinted from Clarke (2005)[27]

Ward et al. (2010) have defined the above surface types as follows:

> "*Cylinder projections* project the surface of the sphere on a cylinder that is put around the sphere. Each point of the sphere is projected outward on the cylinder. Cylinder projection allows the entire spherical surface to be visible...
>
> *Plane projections* are azimuthal projections that map the surface of the sphere to a plane that is tangent to the sphere, with the tangent point corresponding to the center point of the projection...
>
> *Cone projections* map the surface of the sphere to a cone that is tangent to the sphere. Degrees of latitude are represented as circles around the projection center, degrees of longitude as straight lines outgoing from this center" (Ward, et al., 2010, p. 211)

Most Web maps (for example, Google and Bing maps) use Mercator type cylinder projection as illustrated in figure 18. (Hazzard 2011; Microsoft 2013; Spatial Reference Org. 2013).

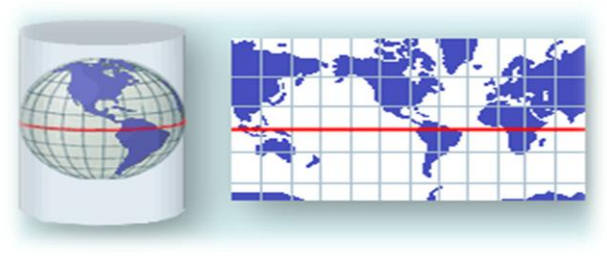


Figure 18: Cylinder projection. The cylinder projection is a surface projection type where a cylinder is wrapped around the Earth with the center of the cylinder's circumference aligned to the equator. Reprinted from Clarke (2005)[28]

The nature of the Mercator type cylinder projection is that there is heavy distortion near the ends of poles. The grid cell gets progressive larger, meaning that the features in high latitudes (the North and South poles) are significantly enlarged. For example, Greenland looks larger than Australia, but in reality it is only about third of the size of it. However it is seen as the best-known cylindrical projection type because the projection is conformal; at any point scale is the same in both directions and the shape of small features is preserved. Ward et al. (2010) have noted that because Mercator is an equal area projection, it is easy to compute and provides a good presentation model of the world maps. (Ward, et al. 2010; Hazzard 2011.)

According to Hazard (2011), there is literately an infinite amount of possible map projections. For example Wikipedia (Wikipedia, 2013) and Radical Cartography (Rankin, 2006) has listed hundreds of different map projections. From a GIS point of view, the ability to spot the nature of map projections is prerequisite. Without the knowledge of projections, the GIS cannot interpret or process selected geodata, such as maps and features, in a right manner (Kraak & Ormeling, 2010).

The projections and projection systems may be very confusing and hard to understand. For example, a common mistake is that the WGS84 and NAD83 are equal projection systems, because they both are used in the GPS devices. However, these systems are not projection systems, but datums describing the geographic coordinate systems of the projection. The NAD83 is referenced to North American Plate, which is “locked in” onto the tectonic plate. The WGS84 is “locked in” to the GPS system. When these two systems were created (1983 and 1984) they were almost identical. However, the Earth is constantly changing, and the tectonic plates are moving, and today the differences between these systems may vary over several meters dependent on the place of the Earth where the measurement is done. One way to tell the nature of the projection is to use a spatial reference system (SRS as discussed in section 2.7.3) along with the data, such as the European Petroleum Survey Group codes (EPSG). The EPSG is a scientific organization involved in oil exploration, which in 2005 was taken over by the International Association of Oil and Gas Producers (OGP). The OGP Geomatics Committee maintains and publishes a dataset of parameters for the coordinate reference system and coordinate transformation description. For example, it provides downloadable EPSG dataset scripts for databases such as MS Access, MySQL, Oracle and PostgreSQL. (Kimerling, 2013; OGP Geomatics Committee, 2013.)

With the help of EPSG and SRS codes the GIS can process the geospatial data in the right manner, because these codes reveals the data projection systems, datums and coordinates shifts the geospatial data is referenced.

3.4.4 Map Scales and Resolution

The scale of the map is the ratio of a distance on the map to the corresponding distance on the ground. In print maps, this ratio is usually presented in somewhere on the map to help a user to translate distances between the real world and the world visualized on the map. For example, if the scale on a map is as illustrated in figure 19, the user knows that 1 inch on the map equals exactly 1 mile on the ground. The map scale ratio is 1:63,500, meaning that one centimeter on the map is 635 meters (63500 centimeters) on the ground.

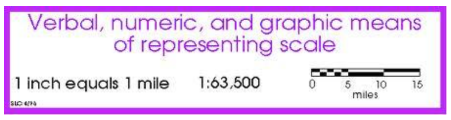


Figure 19: Map scale. In printed map contains usually a verbal, numerical or graphical representation of the map scale, which user may use to translate distances between the map and real world. Reprinted from University of Colorado (1996)

However, determining the exact scale (or resolution) of a particular map image on computer screen is more complicated as the scale depends on several factors. For example, the map image scale is further dependent on the user's screen resolution, including the current longitude and altitude viewed.

In order to calculate the map scale on computer screens, the screen resolution, zoom level, and latitude must be known and assumed that the screen resolution is fixed and equal in both x and y directions. Since screen resolution is usually defined in pixels per inch (PPI), it has to be converted into metrics. (Hazzard 2011; Microsoft 2013).

Microsoft (2013) has defined that Microsoft's Bing map scale can be calculated as

$$Map\ scale = 1 : \left(Screen\ Resolution \frac{pixels}{inch} * 156543.04 \frac{m}{pixel} * \frac{\cos(\varphi)}{2\text{^}zoomlevel}\right),$$

where; the *Screen Resolution* is the number of the pixels shown per inches (a typical screen resolution contains 85 PPI). The (**φ**) is the latitude in degrees (-90 to 90), and the constant value (156543.04) is based on the diameter of the Earth and *zoom level* has values from 1 to 19. (Microsoft, 2013.)

Hence for example, the map scale with 85 PPI monitor screen using Microsoft Bing map, where the map extent is zoomed to maximum level to Helsinki Finland coordinates, the scale (changed from inches to meters) is:

$$Map\ scale = 1 : \left(85 \frac{pixels}{inch} * 39{,}97 \frac{inches}{m} * 156543.04 \frac{m}{pixel} * \frac{\cos(60)}{2^{\wedge 19}}\right)$$, and therefore,

$$Map\ scale = \ 1{:}\,507{,}209\ \approx 1{:}\,510.$$

Therefore, 1 centimeter on a computer screen equals to approximately 5 meters in the real world.

The calculation can be verified by opening the Microsoft Bing Map, and viewing the statue of Mannerheim (located in Helsinki in the front yard of Helsinki Art Museum) in the maximum zoom level as illustrated in figure 20.

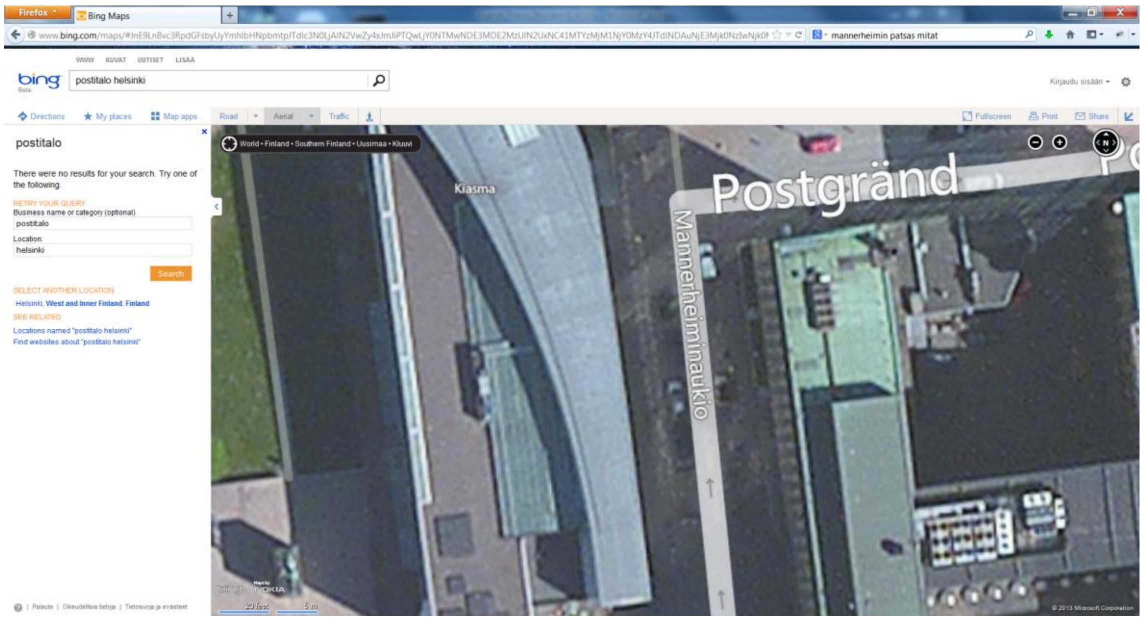

Bing Maps
mannerheimin patsas mitat
bing
postitalo helsinki
Directions
My places
Map apps
Road
Aerial
Traffic
postitalo
There were no results for your search. Try one of the following.
RETRY YOUR QUERY
Business name or category (optional)
postitalo
Location:
helsinki
Search
SELECT ANOTHER LOCATION
Helsinki, West and Inner Finland, Finland
SEE RELATED
Locations named "postitalo helsinki"
Find websites about "postitalo helsinki"
World • Finland • Southern Finland • Uusimaa • Kluuvi
Kiasma
Postgränd
Mannerheiminaukio
20 feet
5 m
© 2013 Microsoft Corporation

Figure 20: The statue of Mannerheim located in Finland at the front yard of Kiasma in the most left-bottom corner. Reprinted from Google Maps (2013)

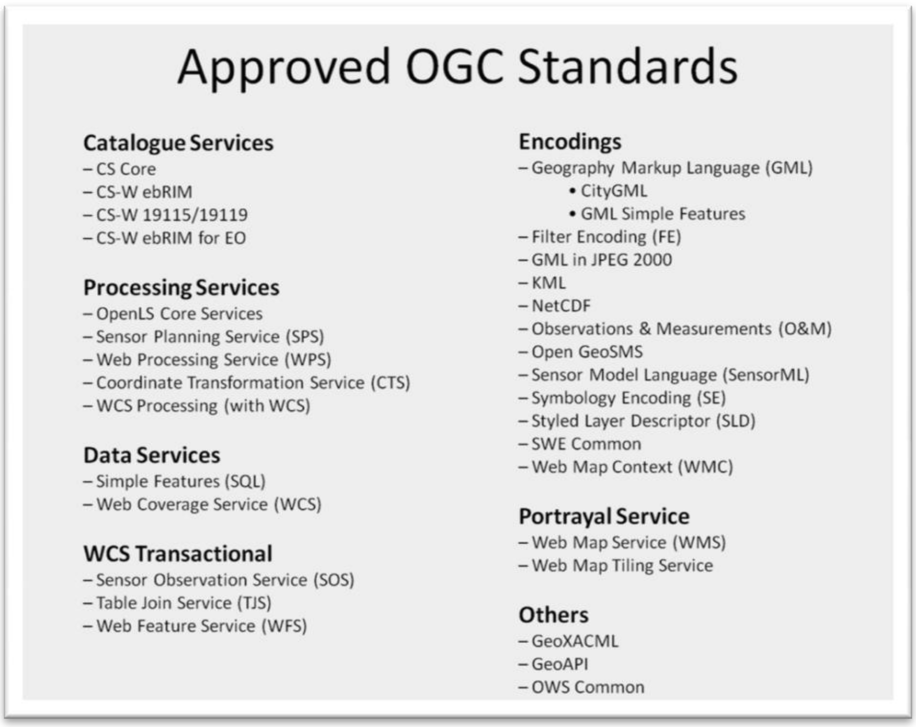
Helsinki Art Museum (Helsinki Art Museum, 2013) has defined that the width of the statue's stand is 2.72 meters, and if the aerial image of the statue is aligned the left-bottom corner with the image scale bar, the calculated scale seems to be approximately correct.

4 Standards and Specifications

4.1 Open Geospatial Consortium

The Open Geospatial Consortium (OGC) is an international industry consortium of 475 participants including companies, government agencies and universities (for example the OGC members from Finland are Vaisala, Geological Survey of Finland and National Land Survey of Finland). The OGC is driving the interoperability between open and commercial GIS software packages provided by many GIS software developers.

The OGC which was founded with eight charter members in 1994 defined a vision of diverse geoprocessing systems communicating directly over networks by means of a set of open interfaces based on the "Open geodata Interoperability Specification" (OGIS). Today the OGC has published 35 adopted standards (see figure 21), which instruct interoperable solutions for "geo-enable" the Web, location-based services and mainstream information technology. (Buehler & Reed 2011; OGC 2013.)



Approved OGC Standards

Catalogue Services
– CS Core
– CS-W ebRIM
– CS-W 19115/19119
– CS-W ebRIM for EO

Processing Services
– OpenLS Core Services
– Sensor Planning Service (SPS)
– Web Processing Service (WPS)
– Coordinate Transformation Service (CTS)
– WCS Processing (with WCS)

Data Services
– Simple Features (SQL)
– Web Coverage Service (WCS)

WCS Transactional
– Sensor Observation Service (SOS)
– Table Join Service (TJS)
– Web Feature Service (WFS)

Encodings
– Geography Markup Language (GML)
  • CityGML
  • GML Simple Features
– Filter Encoding (FE)
– GML in JPEG 2000
– KML
– NetCDF
– Observations & Measurements (O&M)
– Open GeoSMS
– Sensor Model Language (SensorML)
– Symbology Encoding (SE)
– Styled Layer Descriptor (SLD)
– SWE Common
– Web Map Context (WMC)

Portrayal Service
– Web Map Service (WMS)
– Web Map Tiling Service

Others
– GeoXACML
– GeoAPI
– OWS Common

Figure 21: Approved OCS standards. The OGC has published 35 standards, which instructs interoperable solutions for "geo-enable" the Web, location-based services and mainstream information technology. Reprinted from Buehler & Reed (2011)

One of the OGC roles is to provide compliance procedures for GIS software. If the software passes the OGC interoperability test bed and the applicant pays the license fee, the software may use the certification mark granted by the OGC (see figure 22) or the OGC® trademark.

Figure 22: Certification mark. The OGC certificated software may use the OGC compliant logo. Reprinted from OGC (2013)

From the point of view of a GIS architect and Web application developer, the most interesting standards published by the OGC are Simple Features (SQL), Web Feature Service (WFS), Geography Markup Language (GML), Styled Layer Description (SLD) and Web Map Service (WMS) specifications. (Fu & Sun 2011; GeoServer 2013; OGC 2013.)

4.2 EU INSPIRE Directive

The European Community (EC) has entered in force the INSPIRE (Infrastructure for Spatial Information in the European Community) Directive (2007/2/EC) in May 2007.

The Directive goal is to ensure the spatial data infrastructures of the Member States are compatible and usable in a Community. The Directive requires that common Implementing Rules (IR) are adopted in a number of specific areas, such as Metadata, Data Specifications, Network Services, Data and Service Sharing and Monitoring and Reporting areas. (European Commission, 2013)

The OGC follows the work initiated by the Directive and provide INSPIRE stakeholders with an overview of the OGC standards in the INSPIRE clarifications of the IRs with respect to the standards. The OGC has defined that:

> “The INSPIRE Directive Introduces general rules to establish an infrastructure for spatial information in Europe. These rules are related to community environmental policies and policies or activities which impact on the environment. The Directive is taken up and followed by Legally Mandated Organisations (LMOs) operated by the member states and Spatial Data Interest Communities (SDICs) across Europe. The Directive does not require the collection of new spatial data and aims to improve access to and sharing of spatial data held by or on behalf of

> a public authority in Europe. There are 34 spatial data themes laid down in 3 Annexes and the directive entered into force 15 May 2007.
>
> INSPIRE Is a Framework Directive where detailed technical provisions are laid down in implementing rules relating to a number of technical and policy areas, i.e. metadata, interoperability of spatial data sets and services, network services (discovery, view, download, invoke), data and service sharing (policy) and coordination and measures for monitoring and reporting. Once adopted, the Implementing Rules become European legislative acts and national law in 27 Member States and also in some EFTA countries, such as Switzerland and Norway." (OGC, 2012)

For example, the INSPIRE Specification on Coordinate Reference Systems, guidelines that exchanging geospatial data, the ETRS89 (European Terrestrial Reference System 1989) shall be used for the areas within the geographical scope of ETRS89. (European Commission, 2013)

In Finland, the INSPIRE Directive has entered into force through the National Spatial Data Infrastructure Act. According to the Act the National Land Survey of Finland is responsible for the support services required for implementing the Finnish spatial data infrastructure. (Finlex, 2009).

4.3 JHS Recommendations

The National Land Survey of Finland (Development Centre) is hosting the National Geoportal of Finland (Paikkatietoikkuna) at (Maanmittauslaitos, 2012) for displaying spatial data sets and geographic information services. Its goal is to support the creation of spatial data infrastructure and the implementation of the INSPIRE Directive. (National Land Survey, 2012.)

Paikkatietoikkuna maintains the Metadata Catalogue of all open geodata sources, and lists of some most central open spatial data sets and interface services, available in Finland. It also preserves the page containing the current status of the spatial standards and JHS Recommendations.

JHS Recommendations are Public Administration Recommendations, providing information management guidelines for public administration in Finland. JUHTA (Advisory Committee on Information Management in Public Administration) has defined that:

> "A JHS recommendation can be a uniform procedure, definition or instruction to be used in public administration. The JHS system aims to improve the interoperability of information systems and the compatibility of data in them, to facilitate

> cross-sector process development and to make the use of existing data more efficient. The recommendations also aim to minimise overlapping development work, guide the development of information systems and facilitate good common practices in public administration...
>
> The JHS system is focused on:
>
> - Compatibility of information systems and their data and structure
> - Interfaces that increase the interoperability between information systems
> - Data security and privacy protection
> - Cost-effective utilization of information systems" (JUHTA, 2013)

According to Paikkatietoikkuna (2013), there are 6 JHS Recommendations for spatial information listed in more detail at http://www.paikkatietoikkuna.fi/Web/fi/standardit-ja-suositukset.

5 Empirical Part of the Study

This chapter reports the empirical practices performed in this master’s thesis. The goals of the empirical practices were to support the theoretical studies made in this master’s thesis by deepening the practical knowledge behind the subject. The idea was to build a test Web GIS environment including a Web map application, to understand the GIS complexity in action as discussed in chapters 2, 3 and 4.

5.1 Test Environment

The test environment encompassed the database, geographic information server, Web server, and client application as discussed in chapter 2. The database was selected to be PostgreSQL with PostGIS add-on. The selection criterion for PostgreSQL was because it is free and supports the OGC specifications for geodatabases. It also has a good documentation with clear examples. The same selection criteria were applied for GeoServer, which was the choice for the geographic information server.

The Web server chosen for the test environment was Apache, which is part of the XAMPP package. Along with the GIS environment, two Web map client applications were programmed by using a mixture of HTTP and JavaScript code. The first application was a simple Web map application used to practice and demonstrate how predefined spatial data was exercised within the system. This practice and results are described in section 5.2

The second application was a more complex Web map application than the first one, and was named to be Amusement Park by creating some theme and scope for the practice. The Amusement Park application was programmed to demonstrate how to use basemaps derived from Google with additional WMS overlay map data served from a third party such as the National Land Survey of Finland. In addition to the first application, the program was an exercise to test the data exchange between GeoServer and the Web client using the WFS-T protocol. The initial spatial data for the second application were stored to the PostgreSQL database with the help of the QGIS application. This practice and the results are described in section 5.3

The test environment was built on a single laptop computer. The laptop brand was Dell E6520 with the 32-bit Windows 7 operation system (Enterprise edition). The available central memory in the computer was 4 Giga Bytes. The operating system of the computer guided the selection of application packages chosen for the test environment.

5.1.1 Setup

As overviewed in section 5.1, the test environment was built on so that all required applications were installed, and setup to run on a single machine environment, as illustrated in figure 23.

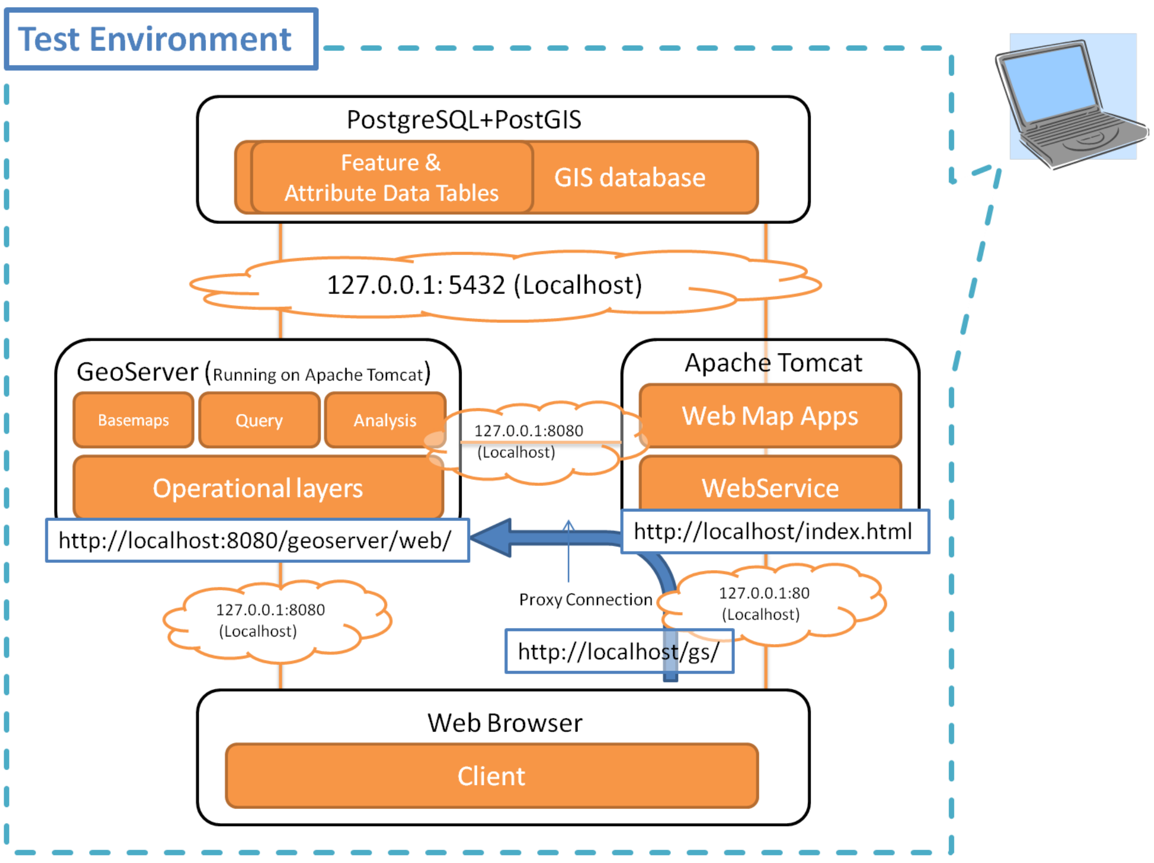

Test Environment
PostgreSQL+PostGIS
Feature & Attribute Data Tables
GIS database
127.0.0.1: 5432 (Localhost)
GeoServer (Running on Apache Tomcat)
Basemaps
Query
Analysis
Operational layers
127.0.0.1:8080 (Localhost)
Apache Tomcat
Web Map Apps
WebService
http://localhost:8080/geoserver/web/
http://localhost/index.html
Proxy Connection
127.0.0.1:8080 (Localhost)
127.0.0.1:80 (Localhost)
http://localhost/gs/
Web Browser
Client

Figure 23: Setup of the test environment.

In the test environment setup, GeoServer served HTTP requests in the TCP/IP port 8080 and the Apache Web server served HTTP requests in the port 80. GeoServer and Apache Web services were hosted by through these ports.

The PostgreSQL database was set (by default) to serve its SQL connections through the port 5432.

5.1.2 Data Flow in the Test Environment

In the test environment (see figure 23), GeoServer and the Apache server were configured to serve the Web client requests as illustrated in figure 24. If the client request URI (Universal Resource Identifier) pointed directly to address *http://localhost:8080*, The GeoServer was responding to that request (figure 24, step 3). If the client request URI pointed to *http://localhost* (or *http://127.0.0.1*), the Apache Web server was responding to that request (figure 24, step 1).

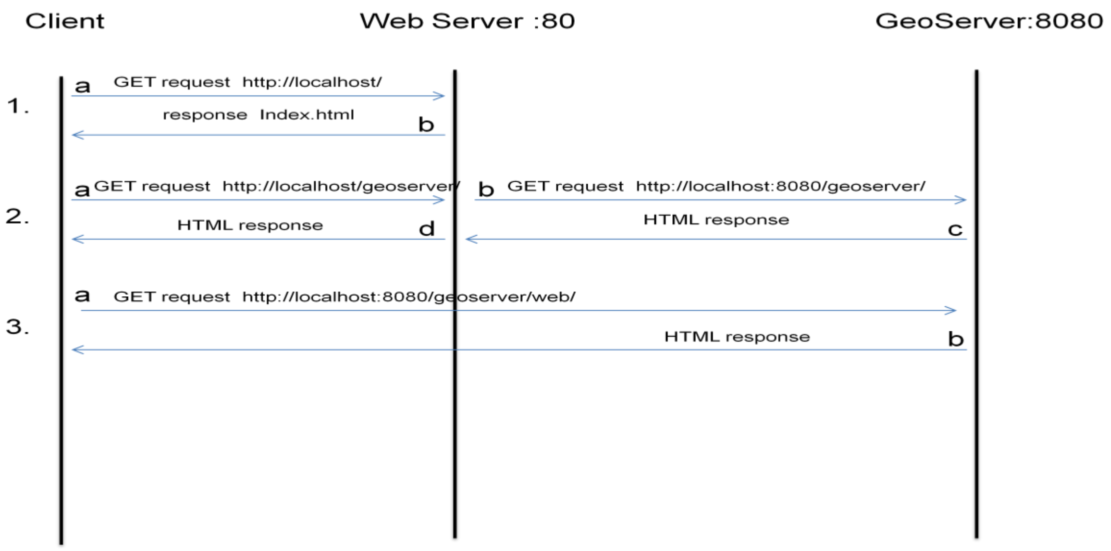

Client
Web Server :80
GeoServer:8080
1.
a GET request http://localhost/
response Index.html b
2.
a GET request http://localhost/geoserver/
b GET request http://localhost:8080/geoserver/
HTML response c
HTML response d
3.
a GET request http://localhost:8080/geoserver/web/
HTML response b

Figure 24: The http data flow examples.

However, if the request made by the client contained the *http://localhost/geoserver/* prefix, the request was intermediated by the Web server to GeoServer (figure 24, step 2), using the *http://localhost:8080/geoserver/* prefix address (the proxy request served by the Web server). The prefix exchange between the Web server and GeoServer was dependant of the Web service root address set on GeoServer.

The objectives behind this setup were to simulate and understand the GIS environment and how GeoServer Web services can be hidden from the Web client and to be served transparently through the Apache Web server, using the server’s proxy capabilities. The test environment setup was also based on the principles discussed in chapter 2.

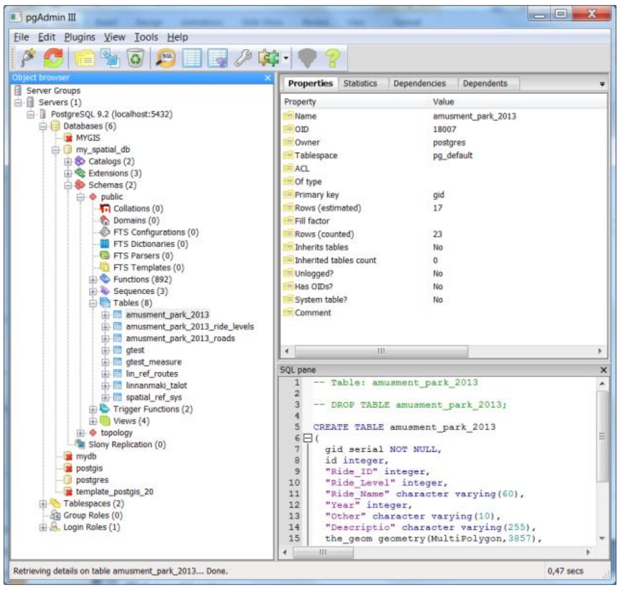
5.1.3 Installation Guide Lines and Tools

The software packages for the test environment were installed as guided on the internet for PostgreSQL at http://www.postgresql.org/ and PostGIS at http://postgis.net/, Java SDK at http://www.oracle.com/technetwork/java/javase/downloads/index.html and XAMP at http://www.apachefriends.org/en/xampp-windows.html. The GeoServer installation and setup process is guided on appendix 2.

Tools

The other tools used in this case study were the pgAdmin for PostgreSQL, Windows Command Prompt, Notepad++, Komodo Edit 7, QGIS and Mozilla Firefox Web browser.

The pgAdmin tool (figure 25) is a graphical user interface application to administrate the PostgreSQL database and is provided within the PostgreSQL package. The pgAdmin tool was used to examine the database information and processes. The tool was also used to monitor the SQL traffic between GeoServer and the database to understand how GeoServer was interpreting data, such as, how the WFS traffic were changed to SQL clauses.



```
Server Groups
  Servers (1)
    PostgreSQL 9.2 (localhost:5432)
      Databases (6)
        MYGIS
        my_spatial_db
          Catalogs (2)
          Extensions (3)
          Schemas (2)
            public
              Collations (0)
              Domains (0)
              FTS Configurations (0)
              FTS Dictionaries (0)
              FTS Parsers (0)
              FTS Templates (0)
              Functions (892)
              Sequences (3)
              Tables (8)
                amusment_park_2013
                amusment_park_2013_ride_levels
                amusment_park_2013_roads
                gtest
                gtest_measure
                lin_ref_routes
                linnanmaki_talot
                spatial_ref_sys
              Trigger Functions (2)
              Views (4)
            topology
          Slony Replication (0)
        mydb
        postgis
        postgres
        template_postgis_20
      Tablespaces (2)
      Group Roles (0)
      Login Roles (1)
```

Property	Value
Name	amusment_park_2013
OID	18007
Owner	postgres
Tablespace	pg_default
ACL	
Of type	
Primary key	gid
Rows (estimated)	17
Fill factor	
Rows (counted)	23
Inherits tables	No
Inherited tables count	0
Unlogged?	No
Has OIDs?	No
System table?	No
Comment	

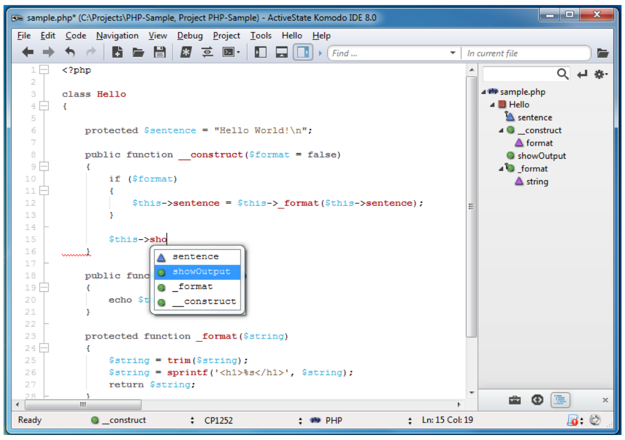
```
-- Table: amusment_park_2013

-- DROP TABLE amusment_park_2013;

CREATE TABLE amusment_park_2013
(
  gid serial NOT NULL,
  id integer,
  "Ride_ID" integer,
  "Ride_Level" integer,
  "Ride_Name" character varying(60),
  "Year" integer,
  "Other" character varying(10),
  "Descriptio" character varying(255),
  the_geom geometry(MultiPolygon,3857),
```

Retrieving details on table amusment_park_2013... Done.   0,47 secs

Figure 25: pgAdmin. pgAdmin is a helper tool for to examine and administrate the PostgreSQL database.

Notepad++ is a free source code editor which is supporting several programming languages (Ho, 2013). The editor was used to read, write and edit simple text files and sample codes encountered during the play of this study.

Komodo Edit 7 is a free integrated development editor (IDE) (see figure 26) provided by the company named ActiveState (ActiveState Software Inc., 2013). It was selected the reason that it has a good support of the code intelligence which were helping to autocomplete, call tips and examine the 3rd party JavaScript library methods and properties.


sample.php* (C:\Projects\PHP-Sample, Project PHP-Sample) - ActiveState Komodo IDE 8.0
File Edit Code Navigation View Debug Project Tools Hello Help
Find ...
In current file
sample.php
Hello
sentence
_construct
format
showOutput
_format
string
Ready
_construct
CP1252
PHP
Ln: 15 Col: 19

```
<?php

class Hello
{

    protected $sentence = "Hello World!\n";

    public function __construct($format = false)
    {
        if ($format)
        {
            $this->sentence = $this->_format($this->sentence);
        }

        $this->sho
    }

    public func
    {
        echo $t
    }

    protected function _format($string)
    {
        $string = trim($string);
        $string = sprintf('<h1>%s</h1>', $string);
        return $string;
    }
```

```
sentence
showOutput
_format
__construct
```

Figure 26: Komodo Edit 7. Komodo Edit is free integrated development editor (IDE) provided by ActiveState.

QGIS is a free and Open Source desktop geographic information tool provided by the GQIS organization (QGIS Org., 2013). QGIS was used to create and analyse the initial geospatial data for the Amusement Park Web application, stored into the database.

5.1.4 Configuring the Apache Proxy Behaviour

After the test environment was installed (see section 5.1.3) the Apache proxy feature was configured as discussed in section 5.1.2. This was done by configuring the Apache *httpd.conf* file as instructed at (Apache Org, 2013). The *httpd.conf* file can be found under the *c:/xamp/apache/conf* directory tree (assuming the XAMP software package was installed as instructed in section 5.1, and the home directory was set during the installation process to be *c:/xamp*) as illustrated in figure 27.

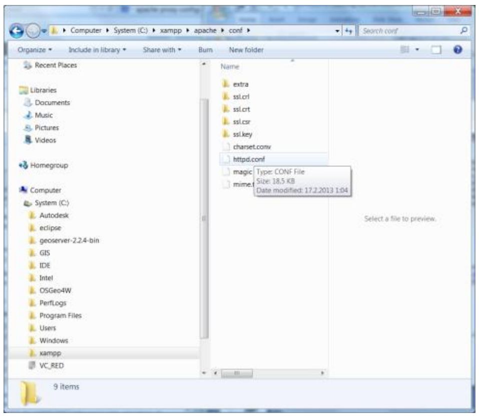


Figure 27: Httpd.conf. The proxy feature was configured by configuring the apache httpd.conf configuration file.

The *httpd.conf* file was configured by using Notepad++ and adding lines 7 to 15 just below the `Listen` directive (line 6.) as illustrated in listing 1.

```
1.  # Change this to Listen on specific IP addresses as shown below to
2.  # prevent Apache from glomming onto all bound IP addresses.
3.  #
4.  #Listen 0.0.0.0:80
5.  #Listen [::]:80
6.  Listen 80

7.  # set proxy for GeoServer requests
8.  ProxyRequests Off
9.  ProxyPreserveHost On
10. <Proxy *>
11. Order deny,allow
12. Allow from all
13. </Proxy>
14. ProxyPass /geoserver/ http://localhost:8080/geoserver/web
15. ProxyPassReverse /geoserver/ http://localhost:8080/geoserver/web
```

Listing 1: Lines 7 to 15 were added to httpd.conf.

Then, the proxy modules were enabled by uncommenting (removing the '#' character from the start of the lines) the following `LoadModule` directives as listed in listing 2 in the httpd.conf.

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

Listing 2: Apache proxy modules. All LoadModule directives, starting with the prefix 'proxy_' were uncommented in the httpd.conf file.

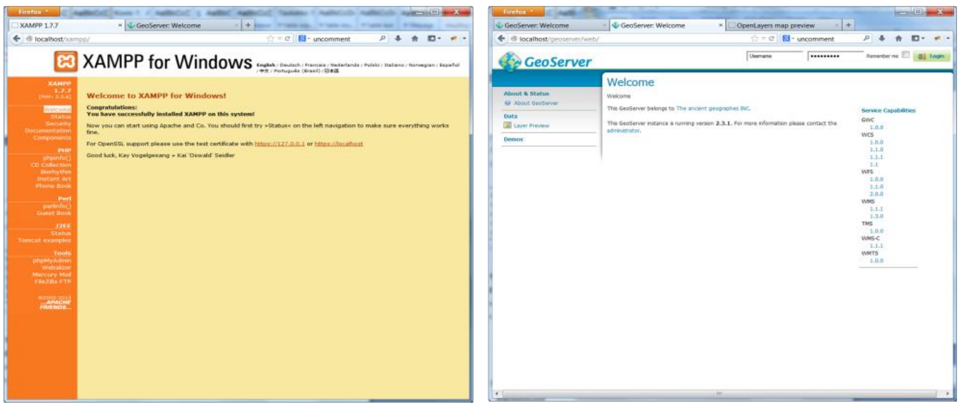
After this, the Apache Web server was restarted to reload new configuration parameters to activate the proxy behaviour.

5.1.5 Testing the Setup

The Web server, GeoServer and the proxy setup were tested by opening the Web browser (Mozilla Firefox) and typing three different URI addresses as follows:

```
1. http://localhost/
2. http://localhost/geoserver/
3. http://localhost:8080/geoserver/
```

The first link address opened the default XAMP for windows Welcome page served by the Web server as illustrated on the left side on figure 28.



Welcome to XAMPP for Windows!

Congratulations:
You have successfully installed XAMPP on this system!

Now you can start using Apache and Co. You should first try »Status« on the left navigation to make sure everything works fine.

For OpenSSL support please use the test certificate with https://127.0.0.1 or https://localhost

Good luck, Kay Vogelgesang + Kai 'Oswald' Seidler

Welcome

Welcome

This GeoServer belongs to The ancient geographies INC.

This GeoServer instance is running version **2.3.1**. For more information please contact the administrator.

Default web Server page using address address http://localhost/

Proxied GeoServer page through the web server using address: http://localhost/geoserver/

Figure 28: Resulting pages after the link tests.

The second link address opened the GeoServer "Welcome" page, intermediated (because of the proxy setup made in httpd.conf) by the Web server from GeoServer. The third address link opened the same GeoServer “Welcome” page, however in this time, directly from GeoServer.

The PostgreSQL and PostGIS setup was tested by typing the following SQL command in the PSQL console (plugin in PostgreSQL pgAdmin III application):

```
SELECT name, default_version, installed_version FROM pg_available_extensions WHERE name
LIKE ‘postgis%’;
```

As the result the lines below was prompted out by proving that the installation went right:

```
       name       | default_version | installed_version
------------------+-----------------+-------------------
 postgis          | 2.0.1           | 2.0.1
 postgis_topology | 2.0.1           | 2.0.1
(2 rows)
```

5.1.6 Summary

This part of the case study presented how the Web GIS test environment was built on using the open source components such as the Apache Web server, GeoServer and PostgreSQL database. It demonstrated how the proxy behaviour is enabled for the Apache Web server. The installation and setup processes for used open source components were quite straight forward and easy. The proxy behaviour demonstrated in this case was not mandatory, but it was done because it presented the very convenient way to hide the unnecessary Web services offered by GeoServer. The proxy behaviour is also simplifying the URI handling process in the client application side; because all URI connections use the same prefix address and TCP/IP port (http://localhost/, and the default HTTP port 80).

The GeoServer installation and start up process (see appendix 2) reveals that the GeoServer is, in fact, a Web server. It uses the Jetty Web server engine which provides the HTTP server and Servlet container, capable of serving a static and dynamic content either from a standalone or embedded instantiations (Eclipse Org., 2013). After this part of the empirical study, the test environment was assembled, and ready for the further tests.

5.2 Creating a Simple Web Map Application

5.2.1 Background and Objectives

The goal of this practical study was to test how the test environment is to be used and to learn a simple Web map application which was created by using the HTML and OpenLayers JavaScript Library with conjunction of the Google Maps, GeoServer and PostgreSQL.

As discussed in chapter 3, the Web GIS application is a Web application presenting some geospatial data as in separate data layers over the basemap. In this practical study, for the simplicity, the application was decided to be programmed with the JavaScript and OpenLayers. OpenLayers is well documented JavaScript library to create map applications. It supports the most OGC GIS Web service standards to exchange and manipulate the spatial data served from the standard Web-enabled GIS servers as discussed in chapters 2 and 4.

The programming was conducted by studying the examples written in the OpenLayers 2.10 Beginners Guide (Hazzard, 2011), and in online manuals of the GeoServer (GeoServer, 2013) and PostgreSQL (PostgreSQL Global Development Group, 2013).

The geospatial data used in the test application was presenting a randomly selected biking route gathered by a Garming GPS sport tracker device. The acquired data was presented in the GPX (Global Positioning eXchange format) data format. The person whom this data was acquired did not reveal any other details included in the data, except for the place where the measuring was done.

The objectives were to store the GPX data in the database, publish it on using GeoServer, and program a Web application which presents the data over the Google's map. From the study point of view, the objective was to understand how to create a simple Web map application portraying the spatial data stored in the database using GeoServer.

5.2.2 Uploading the GPX Formatted Data to the Database

By studying the GPX file format, using the Internet, it was found that the GPX (`.gpx`) format is a light weight XML data format for the interchange of GPS data between applications and Web services (TopoGrafix, 2013). However, the PostgresSQL database does not support any means to insert the GPX formatted data into the database, and therefore, the GPX data had to be converted first into some supported format. This format was discovered to be an ESRI shape file format (`.shp`) by the clue that the PostGIS add-on for PostgresSQL includes the `shp2psql` tool for converting shapefiles into database tables (Open Geo, 2013).

To convert the GPX file to the ESRI shape file, the `gpx2shp` tool made by Toshihiro Hiraoka was used. The `gpx2shp` tool was found at sourceforge.jp Web-site (Hiraoka, 2013).

The biking route data was converted running the `gpx2shp.exe` for the `.gpx` file as executing the following command:

```
C:\MyConvertToolPath\gpx2shp\bin>gpx2shp.exe biking_route.gpx
```

After the conversion, `gpx2tool` created four new files with different file suffixes as illustrated in figure 28.

```
C:\Temp>dir
 Volume in drive C is System
 Volume Serial Number is 2EBD-77E2
 Directory of C:\Temp
27.10.2013  19:05    <DIR>          .
27.10.2013  19:05    <DIR>          ..
27.01.2013  16:41           803 671 biking_route.gpx
27.10.2013  19:05               105 biking_route_meta.txt
27.10.2013  19:05             1 531 biking_route_trk.dbf
27.10.2013  19:05            41 340 biking_route_trk.shp
27.10.2013  19:05               108 biking_route_trk.shx
               6 File(s)        846 755 bytes
               2 Dir(s)  409 598 652 416 bytes free
```

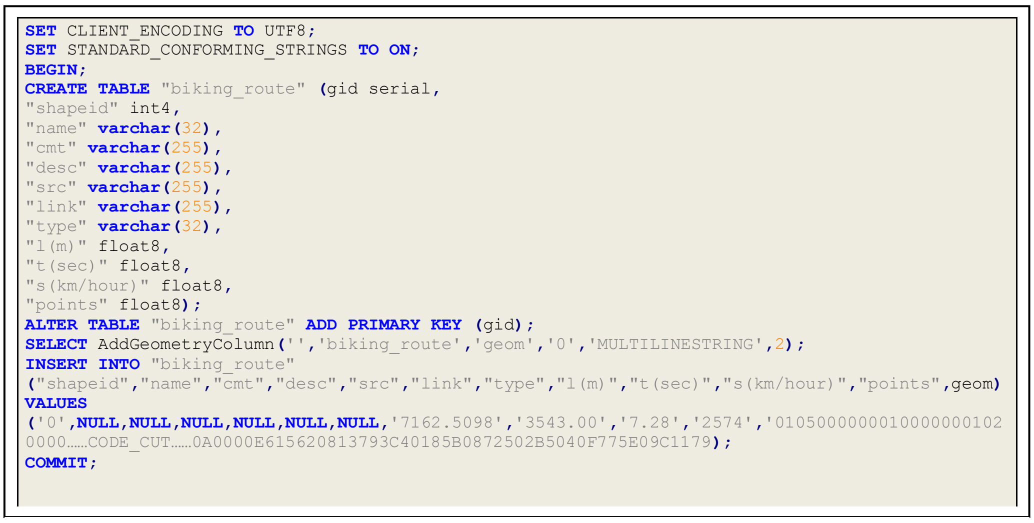
Figure 28: Output of the gpx2shp. The gpx2shp.exe creates four new files from the biking_route.gpx file.

From these new files, the file name with `.shp` suffix was re-converted to SQL-file format executing the PostGIS utility tool `shp2pgsql` as:

```
C:\Program Files\PostgreSQL\9.2\bin>shp2pgsql.exe C:\Temp\biking_route_trk.shp
biking_route > biking_route.sql
```

This command created a new file named as `biking_route.sql`. This file was containing the SQL instruction to be used to create a new data table into the database.

The data table name was specified in the second argument given for `shp2pgsql` tool. The name was set to be “*biking_route*”. The resulting SQL file is illustrated in figure 29.



```
SET CLIENT_ENCODING TO UTF8;
SET STANDARD_CONFORMING_STRINGS TO ON;
BEGIN;
CREATE TABLE "biking_route" (gid serial,
"shapeid" int4,
"name" varchar(32),
"cmt" varchar(255),
"desc" varchar(255),
"src" varchar(255),
"link" varchar(255),
"type" varchar(32),
"l(m)" float8,
"t(sec)" float8,
"s(km/hour)" float8,
"points" float8);
ALTER TABLE "biking_route" ADD PRIMARY KEY (gid);
SELECT AddGeometryColumn('','biking_route','geom','0','MULTILINESTRING',2);
INSERT INTO "biking_route"
("shapeid","name","cmt","desc","src","link","type","l(m)","t(sec)","s(km/hour)","points",geom)
VALUES
('0',NULL,NULL,NULL,NULL,NULL,NULL,'7162.5098','3543.00','7.28','2574','0105000000010000000102
0000……CODE_CUT……0A0000E615620813793C40185B0872502B5040F775E09C1179);
COMMIT;
```

Figure 29: SQL file. The shp2pgsql command file script created the SQL instruction file from the given shape file.

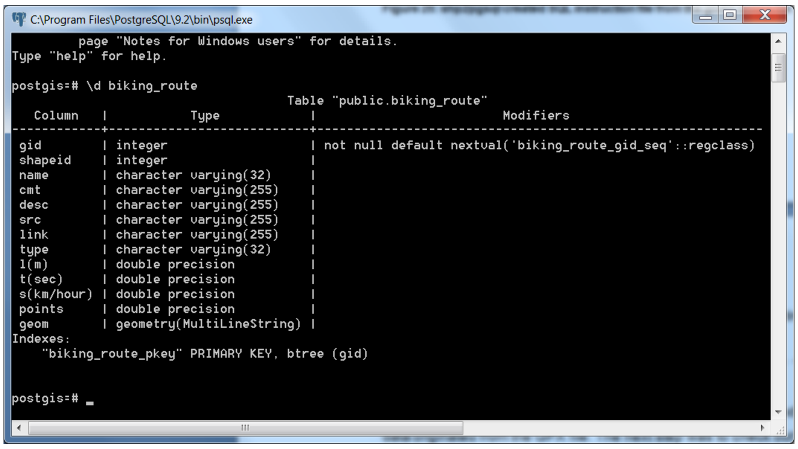
This SQL instruction file was then loaded into the database by executing the following command:

```
C:\Temp>C:\Program Files\PostgreSQL\9.2\bin\psql.exe -U postgres -d postgis -f
C:\Temp\biking_route.sql
```

By the result of this executed command, the database was loaded with the biking route data originated from the GPX file. The next step was to check out that the biking_route table was created with the data, using the pgAdmin tool’s PSQL Console Plugin, by writing the following command, within the console:

```
\d biking_route
```

The console output verified that data table was created (as illustrated in figure 30).



```
        page "Notes for Windows users" for details.
Type "help" for help.

postgis=# \d biking_route
                                     Table "public.biking_route"
   Column    |          Type           |                       Modifiers
-------------+-------------------------+-----------------------------------------------------
 gid         | integer                 | not null default nextval('biking_route_gid_seq'::regclass)
 shapeid     | integer                 |
 name        | character varying(32)   |
 cmt         | character varying(255)  |
 desc        | character varying(255)  |
 src         | character varying(255)  |
 link        | character varying(255)  |
 type        | character varying(32)   |
 l(m)        | double precision        |
 t(sec)      | double precision        |
 s(km/hour)  | double precision        |
 points      | double precision        |
 geom        | geometry(MultiLineString) |
Indexes:
    "biking_route_pkey" PRIMARY KEY, btree (gid)


postgis=# _
```

Figure 30: The PSQL Console output.

As a final check, it was verified that the data table was populated with data by executing the following command within PSQL console:

```
SELECT * FROM biking_route;
```

The command result verified that some data existed in the table.

5.2.3 The SRID of Biking Route Data

A Spatial Reference System Identifier is a unique value to identify the coordinate system the spatial data is presented as discussed in section 2.6. The SRID value of biking route spatial data was originally set to the value 0 (see figure 29, the `AddGeometry statement`), which is a special value, not pointing to any real projection system. The SRID value of biking route geometry was verified by executing following command within PSQL console:

```
SELECT ST_SRID(geom) FROM biking_route;
```

This value had to be changed into the correct SRID value to point out in which coordinate system the biking_route geometry data values are given.

Selecting the right SRID value, at this point, for the geometry data was a pure guess, made by two assumptions; the originated GPX file is presenting GPS data, which is usually presented in WGS84 coordinate system format, and how the route data was formulated in the GPX file.

The SRID value for WGS84 is 4326. This new value was set by executing the following command:

```
SELECT UpdateGeometrySRID(‘biking_route’,’geom’,4326);
```

As the result, the SRID value was set to this new value.

5.2.4 Publishing the Biking Route Data in GeoServer

The next step was to publish the biking route data using GeoServer. This was done by connecting and logging into the administration Web page on GeoServer as instructed on appendix 2, steps 17 and 18. The required configuration and attribute parameters to publish the database information were done by following the instructions given on

GeoServer online user manual at http://docs.geoserver.org/. GeoServer configuration parameters set on the layer publishing process is reported in appendix 3.

As summarized, the biking_route table data was published by conducting the workflow steps as abstracted in figure 31, to following recorded main steps: First, a new workspace was created to present the intended data sources. Second, a new store was created and connected to the database with type of PostGIS and associated to the newly created workspace name. Third, a new layer, exposing the database table columns, was created. The new layer was associated to the workspace and data store name created in the first and the second step. After the third step, the data layer was published and tested, using the GeoServer Layer Preview page.

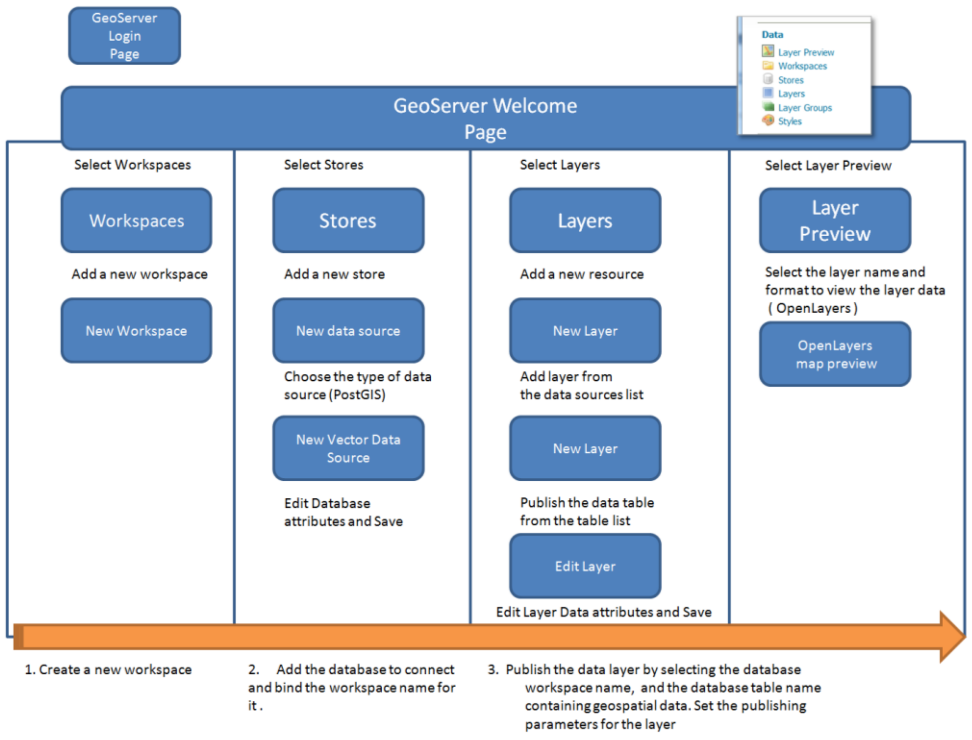
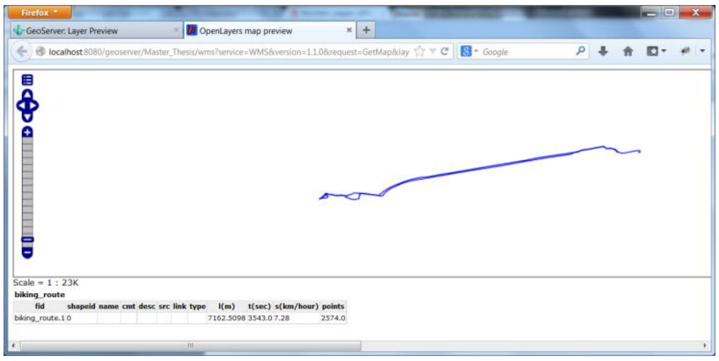

GeoServer Login Page
Data
Layer Preview
Workspaces
Stores
Layers
Layer Groups
Styles
GeoServer Welcome Page
Select Workspaces
Workspaces
Add a new workspace
New Workspace
Select Stores
Stores
Add a new store
New data source
Choose the type of data source (PostGIS)
New Vector Data Source
Edit Database attributes and Save
Select Layers
Layers
Add a new resource
New Layer
Add layer from the data sources list
New Layer
Publish the data table from the table list
Edit Layer
Edit Layer Data attributes and Save
Select Layer Preview
Layer Preview
Select the layer name and format to view the layer data (OpenLayers)
OpenLayers map preview
1. Create a new workspace
2. Add the database to connect and bind the workspace name for it .
3. Publish the data layer by selecting the database workspace name, and the database table name containing geospatial data. Set the publishing parameters for the layer

Figure 31: Abstracted workflow. The abstracted workflow steps to publish a data layer on the GeoServer.

As the result of the biking_route layer preview (using OpenLayers as Common Formats), the page presenting the biking_route table data was portrayed in a graphical form (feature geometry data) as illustrated in figure 32. The preview window also presented other table data included in the biking route geometry when selected.



localhost:8080/geoserver/Master_Thesis/wms?service=WMS&version=1.1.0&request=GetMap&lay

Scale = 1 : 23K

biking_route

fid	shapeid	name	cmt	desc	src	link	type	l(m)	t(sec)	s(km/hour)	points
biking_route.1	0							7162.5098	3543.0	7.28	2574.0

Figure 32: Biking route. The biking route data was previewed using GeoServer Layer Preview feature.

The conclusion at this point was that the biking route data was stored in the database correctly, and GeoServer was able to serve the stored data as a published layer. The GeoServer Preview Layer functionality is a good way to check that the publishing process went right and what kind data are stored in the database table. The layer preview also verifies that the database data can be served through GeoServer in many forms, such as WMS and WFS.

5.2.5 The Biking Route Web Map Application

GeoServer offered a good way to check that the published layer data did contain some data. However, the data previewed was only containing the geometry of data in graphical form with no map context included. Therefore, to show the layer data over some map context, a separate map application must be used. So, to view the biking route data over some map context, the following Web page code was programmed:

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <title>The Biking Route</title>
      <script type='text/javascript' src='OpenLayers.js'></script>
      <script 
src="http://maps.google.com/maps/api/js?v=3.2&amp;sensor=false"></script>
      <script type='text/javascript'>
      var map_area;
      var wms;
      var google;
      //Bounding Box Data values in EPSG:4326
      // BBOX X and Y Data values can be retrieved from GeoServer Edit Layer Page
      var x1 = 28.4448333333333; //Min X
      var x2 = 28.4750516666667; // Max X
      var y1 = 64.67236; //Min Y
      var y2 = 64.677355; // Max Y
      var bounds;
      function init() {
          var wgs84_proj = new OpenLayers.Projection("EPSG:4326");
          var google_proj = new OpenLayers.Projection("EPSG:900913");
          // The map object and definitions for Google Map
          map_area = new OpenLayers.Map('map_element',
              {
```

```
23                  maxResolution: 156543.0339,
24                  projection: new OpenLayers.Projection("EPSG:900913"),
25                  displayProjection: new OpenLayers.Projection("EPSG:4326"),
26                  maxExtent: new OpenLayers.Bounds(
27                      -128 * 156543.0339,
28                      -128 * 156543.0339,
29                      128 * 156543.0339,
30                      128 * 156543.0339),
31                  units: 'm',
32              }
33          );
34          google_map = new OpenLayers.Layer.Google(
35                  "Google: tiekartta",
36                  {'sphericalMercator': true},
37                  {transitionEffect: 'resize'}
38              );
39          biking_route = new OpenLayers.Layer.WMS(
40              "Biking route - Tiled",
41              "http://localhost:8080/geoserver/Master_Thesis/wms",
42              {
43 //layer atttribute format correlating to published GeoServer 'Workspace:Layer Name'
44                  layers: 'Master_Thesis:biking_route',
45                  format:"image/png",
46                  styles:"",
47                  transparent: true,
48                  version:"1.1.0"},
49              {
50                  isBaseLayer: false,
51                  opacity: 0.5,
52              }
53 
54          );
55          map_area.addLayers([google_map,biking_route]);
56          bounds = new OpenLayers.Bounds(x1,y1,x2,y2);
57          bounds.transform(wgs84_proj, google_proj);
58          if(!map_area.getCenter()){
59              map_area.zoomToExtent(bounds);
60          }
61      }
62      </script>
63  </head>
64  <body onload='init();'>
65      The Biking Route
66      <div id='map_element' style='width: 500px; height: 500px;'>
67      </div>
68  </body>
79 </html>
```

Listing 3: The Biking Route map application code.

As mentioned in section 5.2.1, the above HTML and JavaScript code was based on examples written on the OpenLayers Beginner Guide (Hazzard, 2011). The coded Web content contained one div element called as ‘`map_element`’ (listing 3, line 66). This div element was the placeholder for the map content processed by JavaScript. The way, how map and map layers were constructed and portrayed on the Web page, were defined between lines 17 and 62. The basemap used in the application was defined to be the Google Maps. This definition was done between lines 34 and 38. To use Google Maps within OpenLayers, the Google Maps API JavaScript library had to be included (line 5). The biking route data was defined to be as transparent overlay picture over Google Map. This was defined between lines 39 and 52. The service request type for the image data was specified to be WMS. The WMS request type and parameters were defined between lines 41 and 45. The given parameters for the request were: Name of the map layer, in line 40. The WMS request URI address, in line 41. The published data table name on GeoServer (Workspace name:Layer name), in line 44. In which image format type the data is served, in line 45. The image transparency settings in line 46 and 51, and which version of WMS is used (in line 48).

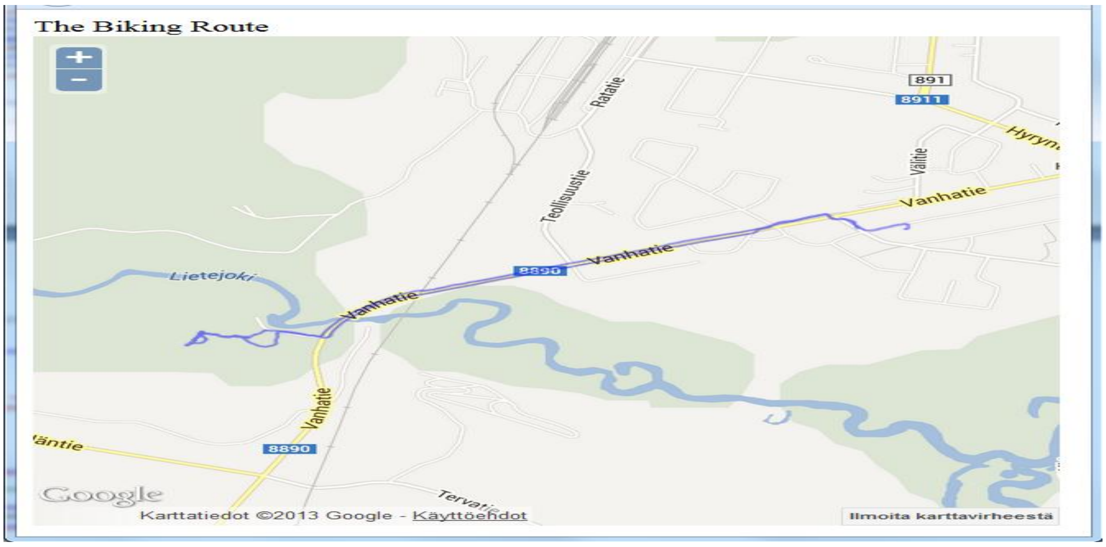


Figure 33: The biking route data was portrayed over Google map.

The result of the Web code is illustrated in figure 33. The biking route data was shown in the blue connected line aligned over the road, named as Vanhatie, pictured by Google Map.

5.2.6 Summary

This practical study contained three main steps: creating and populating the database with geospatial data, publishing the data using GeoServer and programming simple Web map application portraying the geospatial data over Google Maps. The first step was quite straightforward. Converting the acquired biking route data from the GPX format to the ESRI Shapefile format was quite trivial using the documentation found on the Internet. Also, creating and running the SQL instruction file from the Shapefile went with no problems. However, at this point one error was made with Spatial Reference ID. The SQL instruction file created a wrong SRID for the biking route geometry data using the value 0 instead of the right EPSG code value, which was 4326.

Until this small error was found, a huge amount of work was required to find out why the biking route data was not presented over Google Map at the place where it was told to be. This error, however, gave a valuable lesson to understand how important it is to know the projection system of the original spatial data and the projection system of the map used underneath of the data as discussed in chapter 3.

The second step where the database data was published as a WFS layer, via GeoServer, took a while to learn. GeoServer is the reference implementation of the OGC, thus having many features and attribute settings to play with it. Nevertheless, the GeoServer online documentation was very good, and after the basic idea of how the database data was retrieved and published, it was very simple to use. The facility of previewing the data layer as using the OpenLayer format was helping much to understand the naming and attribute conventions used in WFS or WMS requests and in the OpenLayer JavaScript map application.

The third step proved that the OpenLayers as discussed in section 3.2, is a powerful JavaScript Library to create Web map applications in the GIS environment, to view Web-enabled GIS data, with only a few lines of code. This practical study showed that OpenLayers gives a many options for a Web developer to choose how the map application is created. As an example, OpenLayers supports natively the usage of Google Maps, Yahoo Maps, Microsoft Bing Maps or Open Street Maps as discussed in section 2.7. OpenLayers gives also many options to manipulate the Web request parameters to retrieve data in the desired form or protocol.

5.3 Amusement Park Application

5.3.1 Background and Objectives

The goal of this next practical study was to create a bit more complex Web map application called as Amusement Park. The Amusement Park application was programmed to demonstrate how to use basemaps derived from the Google with an additional WMS overlay map data, served by a third party such as National Land Survey of Finland (as discussed in section 3.1). In addition to the first application (see section 5.2), the program was an exercise to test the data exchange between GeoServer and the Web client using the transactional WFS protocol.

The principal difference in between this application and the first application exercised at previous section (5.2) was that the Amusement Park application was also capable to create, delete, modify and store the spatial features played within the application.

As discussed in chapter 3, the Web GIS application is a Web application presenting some geospatial data as in separate data layers over the basemap. The Amusement Park application, encouraged by the first application, was decided to program with the help of OpenLayers. As shown by the biking route practice, to populate the geometry data into the database, some tool or predefined data is required to use in the process. In this practice, the initial spatial data was georeferenced (as discussed in section 3.3.1) and stored in the database whit the help of the Quantum GIS (QGIS) application discussed in section 2.7.

Programming was conducted with the help of examples written in OpenLayers 2.10 Beginners Guide (Hazzard, 2011) and online API documentation of OpenLayers at http://www.openlayers.org/. Other guidelines used to help conduct this practice were found from the online manuals of GeoServer (GeoServer, 2013) and PostgreSQL (PostgreSQL Global Development Group, 2013).

The objectives were to program a Web GIS application which enabled a map view and tools to manipulate spatial features over this map view. Including this, it was also decided to test how the external data can be viewed over the basemap as a separate layer, using standard Web services such as WMS.

5.3.2 Wireframe of the Amusement Park

Concept design layout of Amusement Park was first drawn to give an idea of the application outlook and functions. As the result, the wireframe model, as depicted in figure 34, was created. In the model, the Web page was designed to have four HTML layouts separating different functions of the Amusement Park application. The layouts were a Layer Switcher Box, Map, Info Box and Feature Editing Box.

The Layer Switcher Box was a layout element containing the name of all individual data layers to be switched on or off. With the help of this Layer Switcher, a user may control which layers are overviewed on the map extent on the same time.

The Map layout element was designed to have all the geographical information presented in the map context where the basemap was chosen to be Google Map. Technically the Map layout was also defining the map extent discussed in section 3.2

The Info Box layout element was designed to present a numbered list of the Amusement ride names, which are delivered from the feature data controlled by the map extent.

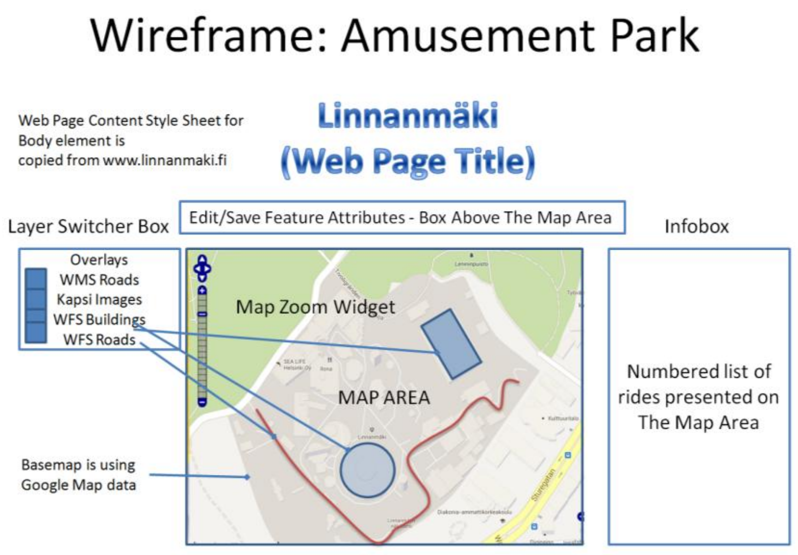

Wireframe: Amusement Park
Web Page Content Style Sheet for Body element is copied from www.linnanmaki.fi
Linnanmäki (Web Page Title)
Layer Switcher Box
Edit/Save Feature Attributes - Box Above The Map Area
Infobox
Overlays
WMS Roads
Kapsi Images
WFS Buildings
WFS Roads
Map Zoom Widget
MAP AREA
Numbered list of rides presented on The Map Area
Basemap is using Google Map data

Figure 34: Amusement Park. A rough wireframe sketch of the Amusement Park Web page

The Feature Editing Box layout was designed to be the display area for tools to play with the feature data displayed on the map.

5.3.3 Database and Data Table Setup

After the concept design plan, the following SQL executions were done to prepare the database for the Amusement Park application; at first, the PostGIS database was connected using the PSQL console. A new geospatially-enabled database was created using the template table:

```
CREATE DATABASE my_spatial_db TEMPLATE=template_postgis_20;
```

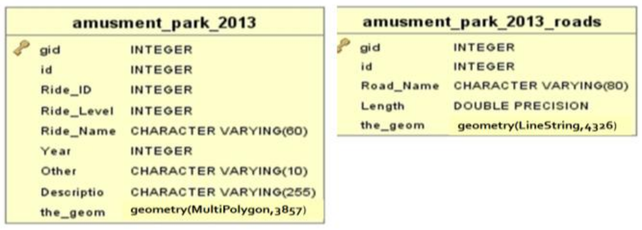
Then, the Amusement Park data table was created with the geometry type column of MultiPolygon which SRID value was set to 3857. (A projected coordinate system based on the WGS84 datum used by Google Maps. The data values are presented in meters):

```
CREATE TABLE amusment_park_2013
(
  gid serial NOT NULL,
  id integer,
  "Ride_ID" integer,
  "Ride_Level" integer,
  "Ride_Name" character varying(60),
  "Year" integer,
  "Other" character varying(10),
  "Descriptio" character varying(255),
  the_geom geometry(MultiPolygon,3857),
  CONSTRAINT amusment_park_2013_pkey PRIMARY KEY (gid)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE amusment_park_2013
  OWNER TO postgres;
```

After this, the Amusement Park Roads table was created with the geometry type column of LineString which SRID value was set to 4326:

```
CREATE TABLE amusment_park_2013_roads
(
  gid serial NOT NULL,
  id integer,
  "Road_Name" character varying(80),
  "Length" double precision,
  the_geom geometry(LineString,4326),
  CONSTRAINT amusment_park_2013_roads_pkey PRIMARY KEY (gid)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE amusment_park_2013_roads
  OWNER TO postgres;
```

The SRID value was referring to the geographic coordinate system based on the WGS84 datum used by the Google Earth.



amusment_park_2013	
gid	INTEGER
id	INTEGER
Ride_ID	INTEGER
Ride_Level	INTEGER
Ride_Name	CHARACTER VARYING(60)
Year	INTEGER
Other	CHARACTER VARYING(10)
Descriptio	CHARACTER VARYING(255)
the_geom	geometry(MultiPolygon,3857)

amusment_park_2013_roads	
gid	INTEGER
id	INTEGER
Road_Name	CHARACTER VARYING(80)
Length	DOUBLE PRECISION
the_geom	geometry(LineString,4326)

Figure 35: Amusement Park data tables. Two data tables were created, one for the rides and one for the roads.

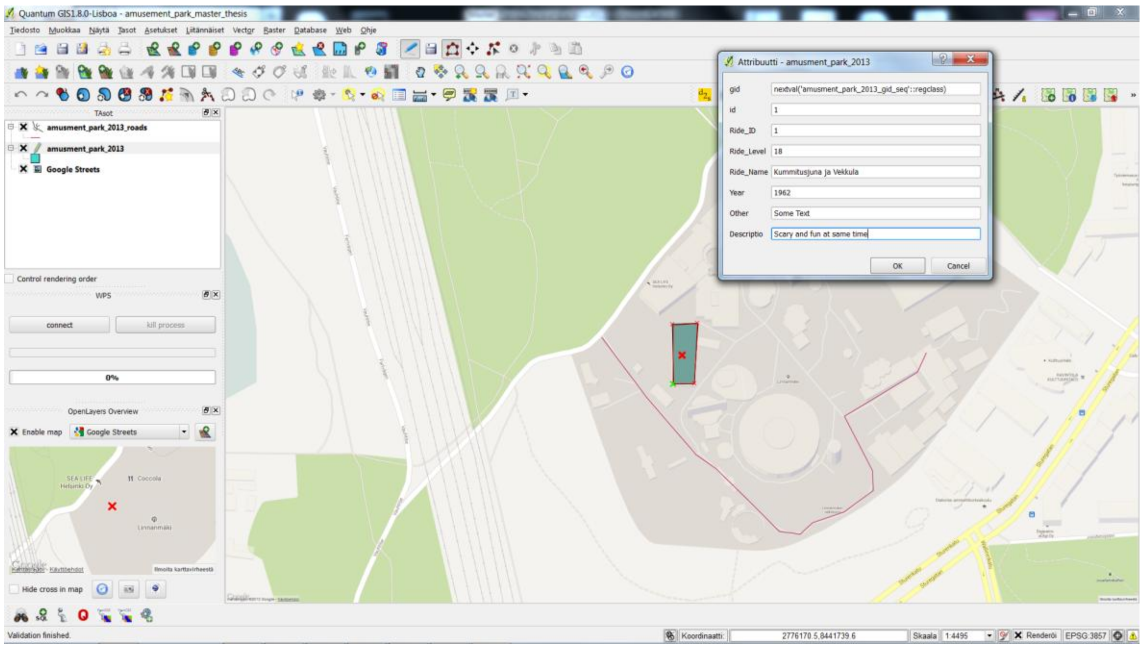
As a result (see figure 35), two data tables were created into the database, one for the “rides” to hold feature data with the geometry data type of MultiPolygon, and one for the roads to hold feature data with the geometry data type of LineString.

5.3.4 Initialising Feature Data Using the QGIS

Before the programming phase, some feature data were initialized into two tables created. The reason for this was to have some geospatial data to test and play within the programming phase. This was done by using the QGIS (see figure 36.) application as follows.

QGIS was connected to the database and data table amusment_park_2013, and amusment_park_2013_roads were added as PostGIS Layers. Both layers CRC (SRID) values were set to use equal values what was set when the data tables were created.

To georeference amusment_park_2013_roads and amusment_park_2013 layers data into the correct location on the Earth, the OpenLayers plugin was installed, using the plugin manager on QGIS.

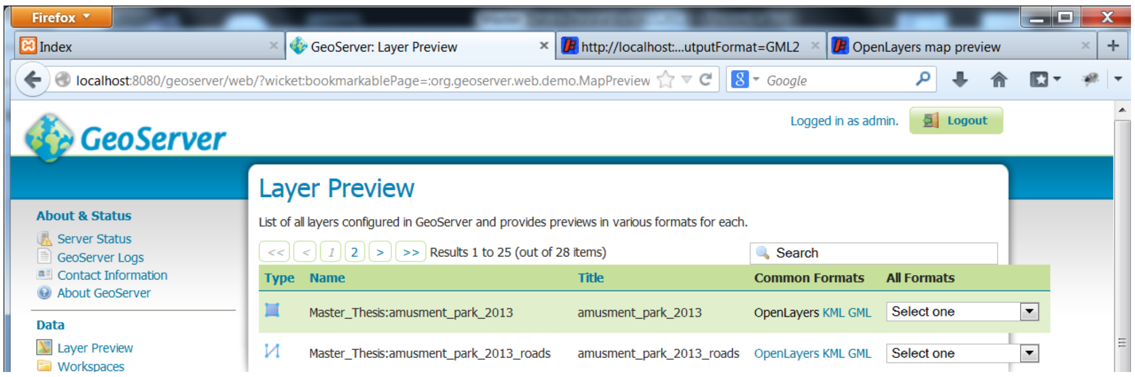


Attribuutti - amusment_park_2013

gid	nextval('amusment_park_2013_gid_seq'::regclass)
id	1
Ride_ID	1
Ride_Level	18
Ride_Name	Kummitusjuna ja Vekkula
Year	1962
Other	Some Text
Descriptio	Scary and fun at same time

Figure 36: QGIS. With the help of QGIS and QGIS OpenLayers plugin, amusement park data tables were added as PostGIS layers and populated with sample feature data. Google Streets map layer was used to georeference the features into the correct location on Earth (Linnanmäki area at Helsinki).

With the help of the plugin, Google Streets map was enabled and added as a map layer. The map view was zoomed and panned to view the area of Linnanmäki Amusement Park at Helsinki Centrum. One road feature was added to the amusment_parks_2013_roads layer and one building (presenting a ride) feature to the amusment_park_2013 layer. At the end of each drawing, the feature attributes were filled in and saved into the database as manifested in figure 36.


Firefox
Index | GeoServer: Layer Preview | http://localhost:...utputFormat=GML2 | OpenLayers map preview

localhost:8080/geoserver/web/?wicket:bookmarkablePage=:org.geoserver.web.demo.MapPreview

GeoServer

Logged in as admin. Logout

About & Status
- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer

Data
- Layer Preview
- Workspaces

Layer Preview

List of all layers configured in GeoServer and provides previews in various formats for each.

<< < 1 2 > >> Results 1 to 25 (out of 28 items) Search

Type	Name	Title	Common Formats	All Formats
	Master_Thesis:amusment_park_2013	amusment_park_2013	OpenLayers KML GML	Select one
	Master_Thesis:amusment_park_2013_roads	amusment_park_2013_roads	OpenLayers KML GML	Select one

Figure 37: Layer Preview. The amusement park application data tables were published in the GeoServer.

To finalize the data initialization, data tables holding the feature data were published and verified in GeoServer as explained in section 5.2.5 and manifested in figure 37.

5.3.5 Amusement Park Web Map Application

The Web application was constructed as illustrated in figure 38, where HTML <header> section contained the JavaScript libraries and Content Style Sheet (CSS) definitions, and the JavaScript code to manage the program logic. The application page layout was defined in the HTML <body> section following the wireframe guidance described in section 5.3.

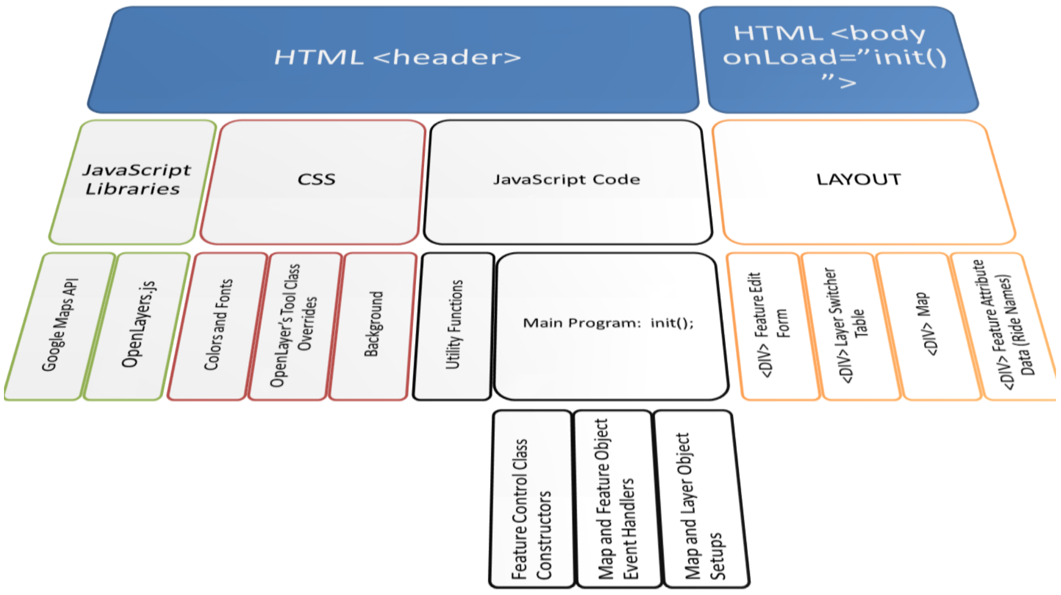

HTML <header>
HTML <body onLoad="init()">
JavaScript Libraries
CSS
JavaScript Code
LAYOUT
Google Maps API
OpenLayers.js
Colors and Fonts
OpenLayer's Tool Class Overrides
Background
Utility Functions
Main Program: init();
<DIV> Feature Edit Form
<DIV> Layer Switcher Table
<DIV> Map
<DIV> Feature Attribute Data (Ride Names)
Feature Control Class Constructors
Map and Feature Object Event Handlers
Map and Layer Object Setups

Figure 38: Key Code Blocks of The Amusement Park application.

The programming was conducted so that first the minimum lines of code were written to get the visible map extent using layers as Google Maps as the basemap and amusement_park_2013 and amusement_park_2013_roads feature data layers served from GeoServer as over map layers using the WFS protocol. This minimum lines of code formed the skeleton file of the application and was similar to the Biking Route application described in section 5.2.6

After this working skeleton file, the programming was iterated step by step including one piece of working code at the time by appending the application behavior and outlook until the application was ready. During these iteration steps, to gain the intended behavior for the application, the OpenLayers Beginner Guide (2011) and examples found on Internet was explored to find the right solution at on each state. The complete Web application contained almost nine hundred lines of code, mostly CSS and program logic code and, on that account, only the key lines of the code are reported as follows in section 5.3.6.

5.3.6 The Key Lines of the Application Code

As the result of the completed Amusement Park application, the key lines of the conducted application code results are herein explained.

To use Google Maps and OpenLayers JavaScript libraries functions, methods and object classes the following two lines had to be defined.

```
    <script type="text/javascript" src='OpenLayers.js'></script>
    <script src="http://maps.google.com/maps/api/js?v=3.2&amp;sensor=false"></script>
```

Then, the HTML page layout was constructed to be a look-a-like of the wireframe design. The layout was constructed using HTML div elements. The `<div id="map">` element were the placeholders for the map extent controlled by the OpenLayers map object.

```
<body onLoad="init()">
    <div id="divModalDialog1" class="divModalDialog">

        <form id="form" oninput="">
            <fieldset>
            <legend><b>Edit Feature Attributes</b></legend>
            Tip: Hover on features over map to select feature wich attributes you want
to edit. Then press Save<br><br>
            <label><b>Ride ID:</b></label>
            <input type="number" id="Ride_ID" name="Ride_ID" required>
            <label><b>Ride Name:</b></label>
            <input type="text" id="Ride_Name" name="Ride_Name" placeholder="Joy Ride"
size=50 required>
            <label><b>Year</b></label>
            <input type="number" id="Year" name="Year" size=4 min=1900 max=2200 re-
quired>
            <label><b>Description</b></label>
            <input type="text" id="Description" name="Description">
            <input type="hidden" id="fid" name="fid">
            </fieldset>
            <fieldset>
            <legend><b>Map Tools</b>
            <table title="Tools">
            <tr><td>Polygon Tools:</td><td><div id="map_tool_panel_polygons"
class="customEditingToolbarPolygons"></div></td>
            </tr>
            <tr><td>Line Tools:</td><td><div id="map_tool_panel_lines"
class="customEditingToolbarLines"></div></td>
            </tr>
            </table>
             </legend>
            </fieldset>
             <input type="button" value="Save" onclick="saveFeatureAttributes()"/><!--
SAVE DATA-->
             <a href="#"><input type="button" value="Close"/></a><!--CLOSE FORM-->
        </form>
    </div>
    <div>
        <table class="table">
        <tr>
            <td>
            <div id="layer_switcher" class="layer_switcher"></div>
            </td>
            <!-- The div id for tool panel and css class as defined earlier-->
            <td>
            <div id="map" class="map">
                Map and Feature Information<div id="messages" class="messages">Info
</div>
```

```
                <div id="divDataForm"><a href="#divModalDialog1"><input type="button"
value="Edit Features"></a></div>
            </div>
            </td>
            <td>
            <div id="information" class="information_data">Info </div>
            </td>
        </tr>
        </table>
    </div>
    <div id="vector_table" style="color:#0000FF; max-height:100%; overflow: auto;">
    </div>
    </body>
</html>
```

The map object was constructed to use the display area which div id was named as "`map`". Other properties set for the map object were used as instructed by the OpenLayers Beginners Guide (2011), to use the correct projections, display resolution and map extent values to work within the Amusement Park data. The object was defined inside the Main Program named as `init()`.

```
// This is where we define the map area and properties for displays units and projec-
tions
    // The map object and definitions for Google Map (which will be our basemap)
    map = new OpenLayers.Map(

        //The html div name where the map is placed
        'map',{
            // The map paramameters
            //To remove all default map controls we need to set empty array
            //The controls for map are set later in this code.
            controls:[],
            // resolution value when using the google maps projection
            maxResolution: 156543.0339,
            // Here we set the default SRID for projection (input proj.)
            projection: new OpenLayers.Projection("EPSG:3857"),

            //This effects how the coordinates are displayed in map view (output proj.)
            displayProjection: new OpenLayers.Projection("EPSG:4326"),
            //Refered Map Max Extent values (corner coordinates) for the Google Maps
            maxExtent: new OpenLayers.Bounds(
                -128 * 156543.0339,
                -128 * 156543.0339,
                128 * 156543.0339,
                128 * 156543.0339
                ),
            //Default display units (meters)
            units: 'm',
        }

    );
```

To have the Google Map as baselayer, the following layer object was constructed:

```
// Basemap setup
    google_map_layer = new OpenLayers.Layer.Google(
        "Google Map",
        {'sphericalMercator': true, minZoomLevel: 3, maxZoomLevel: 20},
        {transitionEffect: 'resize'}
    );
```

The data layer containing Amusement Park "building" features (rides) had more properties to set up than the Google Map layer. The key properties for the layer object were `strategies` and `protocol`. The `strategies` property was defined to use `BBOX`, and `saveStrategy`. Setting the `BBOX` behavior for the layers, was defining that the layer feature data is read from GeoServer when the map view area invalidates some bounds. The `saveStrategy` was an object, which was defined so that all the layer data is automatically saved if the feature data is modified on the layer.

```
saveStrategy = new OpenLayers.Strategy.Save({auto:true});
```

The `protocol` property was set to use WFS protocol version 1.1.0 and use the data served by GeoServer which namespace was amusement_park_2013. This was pointing to the published layer in GeoServer. Including this, also the spatial column name containing the feature geometry data and SRID code had to be defined.

Other properties for the layer object set were style definitions to control how the features are displayed on the layer, and some event listeners to control the feature attribute data.

```
// amusment_park_2013 layer setup (polygon type objects presenting the RIDES)
    amusment_park_2013_layer = new OpenLayers.Layer.Vector(
        "Amusment Park Layer - editable", {

        // How data is loaded / saved between Client and GeoServer
        strategies: [new OpenLayers.Strategy.BBOX, saveStrategy],

        protocol: new OpenLayers.Protocol.WFS(
        {
            version: "1.1.0",
            url: "http://localhost/geoserver/wfs",
            featureNS: "http://localhost/",
            featureType: "amusment_park_2013",
            geometryName: "the_geom",
            srsName:"EPSG:3857",
            schema:
"http://localhost/geoserver/wfs/DescribeFeatureType?version=1.1.0&;typename=amusement_pa
rk:amusement_park_2013",
        }),
        //Documented at http://docs.openlayers.org/library/feature_styling.html
        // and http://dev.openlayers.org/releases/OpenLayers-
2.8/doc/apidocs/files/OpenLayers/Feature/Vector-js.html#OpenLayers.Feature.Vector.style
        styleMap: new OpenLayers.StyleMap({
            'default':{
            strokeColor: "#0000FF",
            strokeOpacity: 0.8,
            strokeWidth: 1,
            fillColor: "#FFFFFF",
            fillOpacity: 0.5,
            pointRadius: 10,
            pointerEvents: "visiblePainted",
            cursor: "(${Ride_Name}\n${Other)",
            // label Ride_ID from the feature attributes
            label : "(${Ride_ID})",

            fontColor: "black",
            fontSize: "10px",
            fontFamily: "Courier New, monospace",
            fontWeight: "bold",
            labelAlign: "cm",
            labelXOffset: "0px",
            labelYOffset: "0px",
            labelOutlineColor: "white",
            labelOutlineWidth: 3
            },
            'select': {
            strokeWidth: 3,
            fontSize:"26px",
            label : "(${Ride_ID})\n${Ride_Name}"

            }
        }),

        // Layer event listeners
        eventListeners: {
            // when map extent contains features in area this event is triggered
            featureadded: function(e) {

            var information_div = document.getElementById('information');
            var newDiv = document.createElement('div');
            var newDivId = e.feature.attributes.Ride_ID;
            newDiv.setAttribute('id',newDivId);
            newDiv.innerHTML = e.feature.attributes.Ride_ID+":
"+e.feature.attributes.Ride_Name+" "+e.feature.attributes.Year;
            information_div.appendChild(newDiv);

            },
            // and vice versa
            featureremoved: function(e) {
            var information_div = document.getElementById('information');
            var olddiv = document.getElementById(e.feature.attributes.Ride_ID);
            information_div.removeChild(olddiv);
            },

            featureselected:function(e) {
            f_attrb = e.feature.attributes.Ride_ID;
            showMsg("selected:" + f_attrb );


            },
        }
        }
    ); // amusment_park_2013_layer definitions ends
```

The Amusement Park road WFS data layer was set similar to the one manifested in the above code of lines. However, to demonstrate the differences between WMS and WFS, another layer of the Amusement Park road feature data was created, but using the WMS protocol.

```
amusment_park_2013_roads_layer_WMS = new OpenLayers.Layer.WMS(
        "WMS Roads",
        // This is an example how the same amusment park road data is requested by using
WMS.
        "http://localhost/geoserver/amusment_park/wms?",

        {
        version: "1.1.0",
        layers: "amusment_park_2013_roads",
        format: "image/png",
        transparent: "true",
        //with WMS the feature styles can be requested from GeoServer and handled using
only one parameter..
        styles  : "YellowRoads"

        },

        {
        isBaseLayer: false

        }
    );
```

The final data layer defined was a test layer which image data was served via WMS by the National Land Survey of Finland.

```
// Test Layer wich retrieves the data from the external source by using the WMS protocol
test_layer = new OpenLayers.Layer.WMS(
        "Kapsi images",
        //More Kapsi portal WMS services at http://kartat.kapsi.fi

//"http://tiles.kartat.kapsi.fi/ortokuva?FORMAT=image/jpeg&VERSION=1.1.1&SERVICE=WMS&REQ
UEST=GetMap&STYLES=&SRS={proj}&WIDTH={width}&HEIGHT={height}&BBOX={bbox}"
        "http://tiles.kartat.kapsi.fi/taustakartta?",

        {
              version: "1.1.1",
              format: "image/png",
              transparent: "true",
        },

        {
              buffer: 1,
              opacity: 0.3,
              alpha: OpenLayers.Util.alphaHack(),
              isBaseLayer: false

        }
    );
```

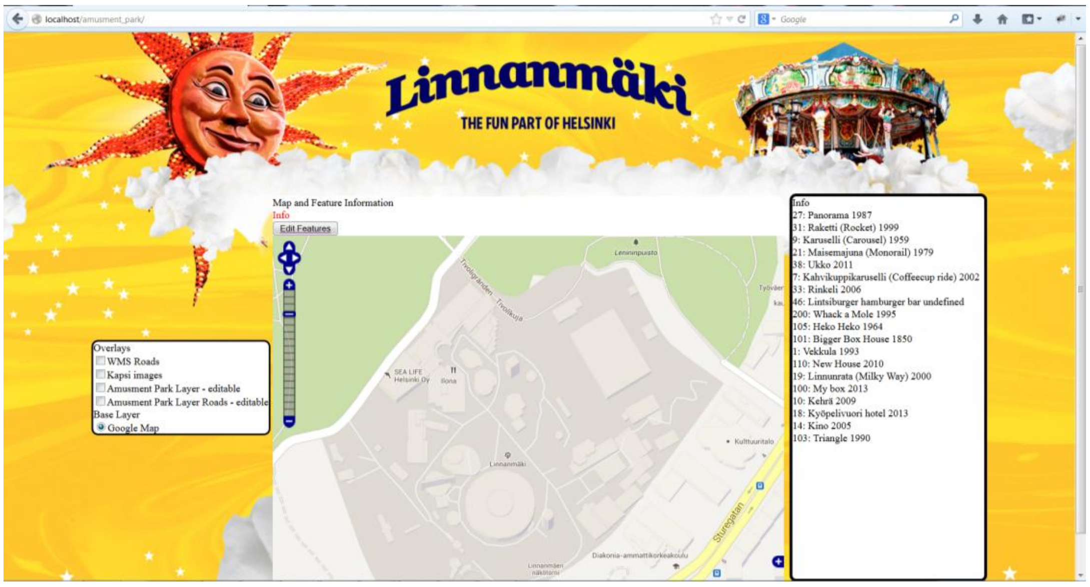
After all the layer definitions, layers objects were added to the map object and the map view was zoomed and focused to display the Linnanmäki Amusement Park area.

```
// Add the layers into the map a
//In the paramemeter list, the order will affect the order how layers are drawn over the
base layer

map.addLayers([google_map_layer,amusment_park_2013_roads_layer,amusment_park_2013_layer,
test_layer,amusment_park_2013_roads_layer_WMS]);

    // Set the Amusement Park center coordinates
    var proj = new OpenLayers.Projection("EPSG:4326");
    var point = new OpenLayers.LonLat(24.94078,60.1884);
    //and zoom to that point
    map.zoomToExtent();

    //set the map extent view to amusement park Lat/Lon coordinates and set the zoom
level for to 14 (it shows Linnanmäki amusement park area nicely)
    map.setCenter(point.transform(proj, map.getProjectionObject()),14);
```

Now the Amusement Park application was fundamentally ready to play with maps and map data as it was planned.

5.3.7 Result

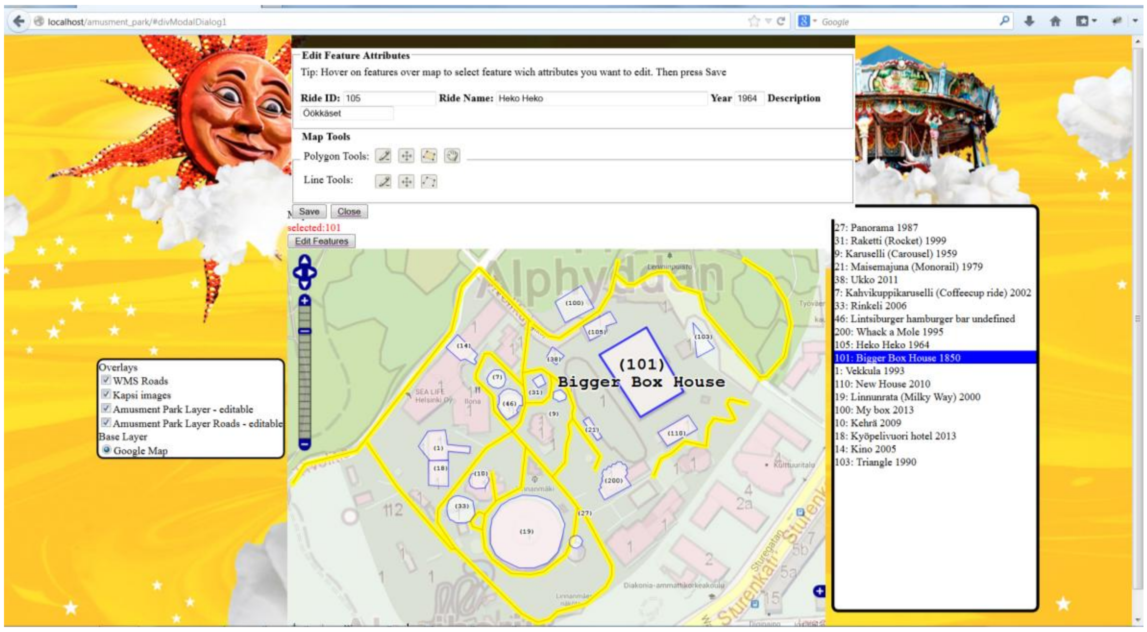
The finalized Amusement Park Web application was capable of displaying the Google Map data and the defined feature data layers on it. The map contained a zoom and pan control handled automatically by the OpenLayers JavaScript library. The fancy background outlook was set by using the original Linnanmäki logo and colors on the Web page. The application was also capable of switching the visibility of all map data layers on or off, using the layer switch box as designed and manifested in figure 39.



Info
27: Panorama 1987
31: Raketti (Rocket) 1999
9: Karuselli (Carousel) 1959
21: Maisemajuna (Monorail) 1979
38: Ukko 2011
7: Kahvikuppikaruselli (Coffeecup ride) 2002
33: Rinkeli 2006
46: Lintsiburger hamburger bar undefined
200: Whack a Mole 1995
105: Heko Heko 1964
101: Bigger Box House 1850
1: Vekkula 1993
110: New House 2010
19: Linnunrata (Milky Way) 2000
100: My box 2013
10: Kehrä 2009
18: Kyöpelivuori hotel 2013
14: Kino 2005
103: Triangle 1990

Figure 39: The Finalized Amusement Park Application.

The info box showed all the ride names presented in the map area, delivered via the amusment_park_2013 WFS data as illustrated in figure 40.



Edit Feature Attributes

Tip: Hover on features over map to select feature wich attributes you want to edit. Then press Save

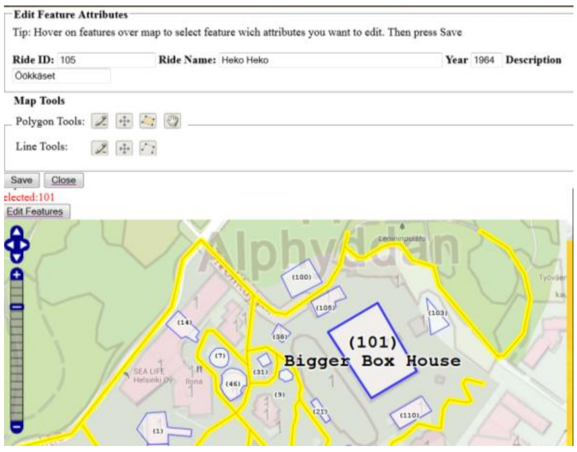
Ride ID: 105 Ride Name: Heko Heko Year 1964 Description Öökkäset

selected:101

27: Panorama 1987
31: Raketti (Rocket) 1999
9: Karuselli (Carousel) 1959
21: Maisemajuna (Monorail) 1979
38: Ukko 2011
7: Kahvikuppikaruselli (Coffeecup ride) 2002
33: Rinkeli 2006
46: Lintsiburger hamburger bar undefined
200: Whack a Mole 1995
105: Heko Heko 1964
101: Bigger Box House 1850
1: Vekkula 1993
110: New House 2010
19: Linnunrata (Milky Way) 2000
100: My box 2013
10: Kehrä 2009
18: Kyöpelivuori hotel 2013
14: Kino 2005
103: Triangle 1990

Figure 40: Amusement Park application. All four layers are switched on to show all data. The visible features in the map area fetched from the amusment_park_2013 ride names are listed in the info box in the right side of the map.

The screenshot presented in figure 41, manifests the Edit feature box, which was opened when the “Edit features” button was pressed. With the Edit box, a user was able to add, modify or delete features and those data attributes per WFS layer by using the map editing tools.



Edit Feature Attributes

Tip: Hover on features over map to select feature wich attributes you want to edit. Then press Save

Ride ID: 105 **Ride Name:** Heko Heko **Year** 1964 **Description** Öökkäset

Map Tools

Polygon Tools:

Line Tools:

Save Close

elected:101

Edit Features

Figure 41: Editing feature attributes. With the Edit box, a user was able to add, modify or delete features and those attribute per WFS layer.

Figure 41 also illustrates well how Kapsi portal data served by National Land Survey of Finland, was portrayed on its own transparent image layer over Google Map. The Kapsi layer added some pink houses and gray paths and roads and place names included over Google Map. The yellow roads are the WMS image data presented in WMS road layer which is based on the same amusment_park_2103_road data presented in vector form in Amusement Park Roads layer (brown thin lines under the yellow lines).

5.3.8 Summary

As the goal of this practical study was to create a bit more complex Web map application called as Amusement Park, the goal was reached as planned.

From a developer point of view, this practice revealed that creating a Web map application in not so complex as one could first think. Constructing a simple Web map application with some data overlays is quite trivial work if the OpenLayers JavaScript library is used. However, for the first timer the most time consuming work is to find and learn all the thousand features provided by the OpenLayers library. Fortunately the OpenLayers is well documented and provides plenty of good examples. What comes to the data exchange between an application and GIS server, using the standard OGC Web services such as WMS and WFS, a developer must understand how the data is requested and served by using these standards. Without this understanding, setting up the correct property values, for example for the WMS or WFS layer objects, can be very difficult to get the layers data work as intended.

During the practical studies, I learned that GeoServer is providing practical methods to test the published data using the Layer Overview feature. With this feature, it was easy to learn how OpenLayers should form the request URI and the Web service parameters. By combining this knowledge to the request data sent by an application (for example using the Firefox debugger), a developer can learn how OpenLayers layer object properties must be set to gain the right request.

I also learned that OpenLayers supports amongst the other Web service protocols the OGC CSW service standard (Catalog Service for Web) discussed in the section 2.5. With the help of the CSW, applications could dynamically add data layers to a user request on the demand. However, implementing this feature for the application was not tested.

The other lesson learned within this practice was the importance to know the coordinate system where the geometry data is referenced. Without this knowledge, to present the data in the proper way in an underlying map cannot be done. The most common mistake during this practice was that the projection system was defined wrongly in the database, GeoServer or in the Web application, which made the geometry data disappear from the map.

Lastly, it was discovered that if a developer starts defining the spatial data from the scratch, he must use some helper application to create a geometry data which is referenced in some coordinate system. The helper application can be self-made as the amusement park application, or QGIS, as demonstrated in this master’s thesis or any other GIS product available in the market. Without the use of a GIS application, storing the spatial data in the database manually (for example, using the SQL insert commands) can be a very time consuming and error-sensitive process.

6 Conclusions

The goal of this master’s thesis was to understand what the Geographic Information System is and what key technologies and fundamentals lie behind this area from the point of view of a developer. The goal was achieved by discovering the professional literatures and though hands-on case study practises presented in this thesis.

This master’s thesis discovered that the GIS is a collection of software, hardware, data and people. However, this collection complies with the general characteristics of any other information system, thus making the GIS, from a developer point of view no different than any other information system. Yet, GIS seems to be thought of as more complex to master than some other information system. The reasons for this kind of thinking might lie behind the history of the GIS technology and the people involved in the technology, and in the goal what it is mainly meant to used for (to visualize the geospatial data).

From the history point of view, the GIS business has been (and still is) dominated by the ESRI. This have had the impact that in most organizations information technology experts still think that the only solution for a spatial information system is to purchase the “ESRI” product because it is offering a solution for “everything” and it is “easy to install”. For an organization with loads of money, this can be the easiest choice, but from a developer point of view the following questions should first be asked: what is the value-add in the selected GIS product for the business, what spatial data is provided and in which form, what are the GIS functions required to do the business analysis and how the end results of analysis are used by others in the business or ecosystem.

It seems that the Web 2.0 era, Google, Bing and other cloud service providers, including open source communities, have opened doors for “common” Web developers to create spatially enabled applications without even understanding the GIS technology itself. The Web 2.0 era has also initiated the technology of Web services where the basic idea is to communicate data between Web applications through the common application protocol interfaces, such as the HTTP protocol. This is one of the key technologies behind the Web GIS, which is one of the flavours of the GIS, the ability to distribute and exchange common data between GIS users and applications. The key standards behind the distributed Web GIS are the OGC standards, which are specifying the common ground for the systems handling the spatial data. From a developer point of view, these standards give guidelines as to what are the GIS-specific Web services, and how these services are distributed and used. These standards are also driven partially by the European Union through the Inspire Directive, which enforces public administrations to open their data collections (open data) for public use.

From a technology point of view, a developer should understand that there is no single system or application to choose which fully comprises the GIS. The standardized GIS encompasses network-connected OCS-compliant data storage, a geoinformation server and an application. The data storage can be a file system or database. The geoinformation server is the core system of any GIS, and its main role is to mediate spatial data between applications and data storage. The spatial analysis, functions and operations, can be run by the choice of the developer in the data storage, geoinformation server, or in the application end. However, this discretion is dependent on the selected GIS components, the total architecture, and the workload of services using the GIS components.

From a geographical information point of view, a developer should understand that in the GIS the data is described through the concept of features. The feature is a geographical object containing some attribute data. The feature is the geographical object only if the object's geometry data is georeferenced to some coordinate system. For example, GeoServer cannot publish data tables, which do not have a column for geometry information. Through the georeferencing, the feature, the spatial data, can be presented over many map types, independent of the map's projection system. However these projection systems must be understood on such a level that the feature data must be projected over the map using the same projection system as the underlying map uses. Therefore, a developer should always know which coordinate system the original spatial data is referenced to and what coordinate system the map where the data is presented is using.

To summarize, the goal of any GIS is to visualize and manage geospatial data over maps. In practice, the most important component of the system is a geoinformation server. On an abstract level, this server can be seen as a publication system for the spatial data sources, as well as a messaging broker which mediates spatial data between applications and data storages. The learning curve to become a GIS professional in all GIS areas can be very long and challenging for the reason that it contains so many technology components.

.

References

ActiveState Software Inc.: *Komodo-edit.* [Online]. 2013
URL: http://www.activestate.com/komodo-edit.
Accessed 27 October 2013.

Apache Friends: *XAMP.* [Online]. 2013.
URL: http://www.apachefriends.org/en/xampp.html.
Accessed 18 October 2013.

Apache Org: *Apache Module mod_proxy.* [Online]. 2013.
URL: http://httpd.apache.org/docs/current/mod/mod_proxy.html.
Accessed 2 May 2013.

Ball, M.: *Spatial Sustain.* [Online]. 2009.
URL: http://www.sensysmag.com/spatialsustain/
reference-for-80-of-data-contains-geography-quote.html.
Accessed 13 March 2013.

Bestebreurtje, J.: *GIS Project Management,* [Online].
Manchester: UNIGIS Amsterdam; 1997.
URL: http://www.feWeb.vu.nl/unigis/downloads/msc/hans%20bestebreurtje.pdf.
Accessed 4 February 2013.

Buehler, G. & Reed, C.: *OGC Orienation Slides.* [Online]. 2011.
URL: https://portal.opengeospatial.org/files/?artifact_id=47735.
Accessed 21 April 2013.

Butler, P.: *Facebook connections.* [Art] (Facebook). 2010.
URL: http://edition.cnn.com/2010/TECH/social.media/12/14/
facebook.relationships.mashable/.
Accessed 10 March 2013.

Caitlin Demsey: *What is GIS.* [Online]. 2012.
URL: http://www.gislounge.com/what-is-gis/.
Accessed 9 March 2013.

Clarke, K. C.: *USCB Geography.* [Online]. 2005.
URL: http://www.geog.ucsb.edu/~kclarke/G176B/Lecture04.ppt
Accessed 15 April 2013.

Cohen, S.: *From Design Into Print: Preparing Graphics and Text for Professional Printing.* 1st ed.
Berkeley: Peachpit Press. 2009.

Connolly, M. & Begg, C. E.: *Database Systems: A Practical Approach to Design, Implementation, and Management.* 4th ed.
Pearson Education Limited. 2005.

Coppock, J. T. & Rhind, D. W.: *Geographic Information Systems - Chapter 2. The History of GIS.*
John Wiley & Sons Ltd. 1991.

DeMers, M.: *GIS for Dummies.*
Indianapolis: Wiley Publishing, Inc. 2009.

Eclipse Org.: *jetty.* [Online]. 2013.
URL: http://www.eclipse.org/jetty/about.php.
Accessed 5 May 2013.

ESRI Support Center: *ArcGIS Server 9.3.1 Help, Geocoding services.* [Online]. 2013.
URL: http://Webhelp.esri.com/arcgisserver/9.3.1/java/index.htm#geocode_service.htm.
Accessed 30 March 2013.

ESRI Support Center: *ArcGIS Server 9.3.1 Help, Geodata services.* [Online]. 2013.
URL: http://Webhelp.esri.com/arcgisserver/9.3.1/java/index.htm#geodata_service.htm.
Accessed 30 March 2013.

ESRI Support: *GIS Dictionary.* [Online]. 2013.
URL: http://support.esri.com/en/knowledgebase/GISDictionary/term/attribute%20query.
Accessed 28 March 2013.

ESRI: *ArcGIS Resource Center - Desktop 10.* [Online]. 2012.
URL: http://help.arcgis.com/en/arcgisdesktop/10.0/help/
index.html#//009t0000000w000000.
Accessed 24 March 2013.

ESRI: *ArcGIS Resource Center, ArcGIS API for JavaScript 1.6.* [Online]. 2013.
URL: http://resources.esri.com/help/9.3/arcgisserver/apis/javascript/arcgis/help/
jshelp_start.htm#jshelp/intro_extents.htm.
Accessed 13 April 2013.

ESRI: *ArcGIS Resource center, Server 10, Network analysis service.* [Online]. 2013.
URL: http://help.arcgis.com/en/arcgisserver/10.0/help/
arcgis_server_dotnet_help/index.html#//009300000055000000.
Accessed 30 March 2013.

ESRI: *What is GIS.* [Online]. 2013.
URL: http://www.esri.com/what-is-gis/overview
Accessed 9 March 2013.

European Commission: *Data Specifications.* [Online]. 2013.
URL: http://inspire.jrc.ec.europa.eu/index.cfm/pageid/2.
Accessed 21 April 2013.

European Commission: *INSPIRE - Infrastructure for Spatial Information in the European Community.* [Online]. 2013.
URL: http://inspire.jrc.ec.europa.eu/.
Accessed 21 April 2103.

FEM Wiki: *Field Epidemiology Manual*
- Choosing an Appropiate Type of Map. [Online]. 2013.
URL: https://wiki.ecdc.europa.eu/fem/w/fem/choosing-an-appropriate-type-of-map.aspx
Accessed 20 April 2013.

Finlex: *Finlex - 421/2009.* [Online]. 2009.
URL: http://www.finlex.fi/fi/laki/alkup/2009/20090421.
Accessed 21 April 2013.

Friendly, M.: *Milestones in the history of thematic cartography, statistical graphics, and visualization.*[Online]. 24 August 2009
URL: http://euclid.psych.yorku.ca/SCS/Gallery/milestone/milestone.pdf.
Accessed 20 February 2013.

Fu, P. & Sun, J.: *Web GIS.* 1st ed.
Redlands(California): Esri Press. 2011.

Gandhi, U.: *Tutorial: Georeferencing Topo Sheets, Topo Maps, Satellite Image or Scanned Maps in QGIS.* [Online]. 2013.
URL: http://qgis.spatialthoughts.com/2012/02/tutorial-georeferencing-topo-sheets.html.
Accessed 14 April 2013.

Georeference Org: *Georefence Org.* [Online]. 2011.
URL: http://www.georeference.org/doc/
manifold.htm#universal_transverse_mercator_utm_.htm.
Accessed 18 April 2013.

GeoServer: *GeoServer User Manual.* [Online]. 2013.
URL: http://docs.geoserver.org/stable/en/user/.
Accessed 21 April 2013.

Google Inc.: *A Brief History of Google Maps.* [Online]. 2012.
URL: http://maps.google.com/help/maps/helloworld/behind/history.html.
Accessed 16 March 2013.

Google Inc.: *Google Maps for Business.* [Online]. 2013.
URL: https://www.google.com/enterprise/earthmaps/legal/us/
maps_purchase_agreement.html.
Accessed 7 April 2013.

Guan, G. F. G.: *Center of Excellence for Geospatial Information Science.* [Online]. 2006.
URL: http://www.geog.ucsb.edu/~guan/courses/176A_Summer06/lecture04.ppt.
Accessed 17 October 2013.

Hazzard, E.: *OpenLayers 2.10, Beginner's Guide.* 1st ed.
Olton, Birmingham: Packt Publishing.

Helsinki Art Museum: *Julkiset veistokset.* [Online]. 2013.
URL: http://www.taidemuseo.fi/suomi/veisto/veistossivu.html?id=179&sortby=statue.
Accessed 20 April 2013.

Hiraoka, T.: *gpx2shp.* [Online]. 2013.
URL: http://gpx2shp.sourceforge.jp/index.html.en.
Accessed 8 May 2013.

Ho, D.: *Notepad++.* [Online]. 2013.
URL: http://notepad-plus-plus.org/.
Accessed 27 October 2013.

Huang, X.B: *Geo-Agent based spatial information services and application intergration.* Phd Dissertation.
Beijing University: 2002.

JUHTA: *JHS - Public Administration Recommendations.* [Online]. 2013.
URL: http://www.jhs-suositukset.fi/Web/guest/jhs.
Accessed 21 April 2013.

Kevin: *Cartography Tools.* [Online]. 2010.
URL: http://mappractical.blogspot.fi/2010/01/cartography-tools.html.
Accessed 14 April 2013.

Kimerling, D. J.: *WGS84 vs NAD83.* [Online]. 2013.
URL: http://mappingcenter.esri.com/index.cfm?fa=ask.answers&q=740.
Accessed 1 December 2013.

Kraak, M.-J. & Ormeling, F.: *Cartography Visualization of Spatial Data.* 3rd ed.
Harlow, Essex: Pearson Education Limited. 2010.

Kralidis, A. T.: *Geospatial Web Services:An Evolution of Geospatial Data Infrastructure.* [Online]. 2005.
URL: http://www.kralidis.ca/gis/masters/thesis/.
Accessed 10 March 2013.

Maanmittauslaitos: *Paikkatietoikkuna.* [Online]. 2012.
URL: http://www.paikkatietoikkuna.fi/Web/fi.
Accessed 21 April 2013.

Martin, G.: *The Phrace Finder.* [Online]. 2013.
URL: http://www.phrases.org.uk/meanings/a-picture-is-worth-a-thousand-words.html.
Accessed 18 April 2013.

Microsoft: *MSDN - Understanding Scale and Resolution.* [Online]. 2013.
URL: http://msdn.microsoft.com/en-us/library/aa940990.aspx.
Accessed 19 April 2013.

Microsoft: *Three-Layered Services Application.* [Online]. 2013.
URL: http://msdn.microsoft.com/en-us/library/ff648105.aspx.
Accessed 24 March 2013.

Miller, W. R.: *Introducing Geodesign: The Concept.*
Redlands: ESRI. 2012.

Muice, A.: *Raster Image Processing Tips and Tricks — Part 1: Georeferencing.* [Online]. 2010.
URL: http://blogs.esri.com/esri/arcgis/2010/10/19/georef1/.
Accessed 14 April 2013.

National Land Survey: *Paikkatietoikkuna.* [Online]. 2012.
URL: http://www.paikkatietoikkuna.fi.
Accessed 21 April 2013.

NAVTEQ Maps: *History.* [Online]. 2012.
URL: http://www.navteq.com/company_history.htm.
Accessed 24 March. 2013.
OGC: *OGC - Making location count.* [Online]. 1994 – 2013.
URL: http://www.opengeospatial.org/standards/wms.
Accessed 28 March 2013.

OGC: *OpenGIS® Web Map Tile Service Implementation Standard.* 1.0.0 ed. 2010.

OGC: *OGC Market Report - Open Standars and Inspire.* 2012.

OGC: *About OGC.* [Online]. 2013.
URL: http://www.opengeospatial.org/ogc.
Accessed 21 April 2013.

OGC: *OGC Standars.* [Online]. 2013
URL: http://www.opengeospatial.org/standards/is.
Accessed 21 April 2013.

OGP Geomatics Committee: *Geomatic - OGP Geomatics Committee.* [Online]. 2013.
URL: http://www.epsg.org/.
Accessed 19 April 2013.

Open Geo: *OpenGeo Suite Library.* [Online]. 2013.
URL: http://suite.opengeo.org/docs/dataadmin/pgGettingStarted/shp2pgsql.html.
Accessed 8 May 2013.

OGC: *Web Feature Service Implementation Specification.* 2005.
Open Geospatial Consortium Inc.

OGC: *OpenGIS® Web Map Server Implementation Specification.* 1.3.0. 2006.
Open Geospatial Consortium Inc.

OGC: *OpenGIS® Catalogue Services Specification.* 2.0.2, Corrigendum 2 Release. 2007.
Open Geospatial Consortium Inc.

OGC: *OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture.* 1.2.1. 2011.
Open Geospatial Consortium Inc.

OpenPlans Org.: *GDAL Image Formats.* [Online]. 2013.
URL: http://docs.geoserver.org/stable/en/user/data/raster/gdal.html#data-gdal.
Accessed 14 April 2013.

OpenStreetMap Org.: *Openstreetmap.* [Online]. 2013.
URL: http://www.openstreetmap.org/.
Accessed 6 April 2013.

OSGEO: *GRASS.* [Online]. 1998-2013.
URL: http://grass.osgeo.org/#.
Accessed 24 March 2013.

OSGeo: *Chapter 4. Using PostGIS: Data Management and Queries.* [Online]. 2013.
URL: http://postgis.net/docs/using_postgis_dbmanagement.html#spatial_ref_sys.
Accessed 6 April 2013.

Papagelis, M.: *Web App Architectures: Multitier (2-tier, 3-tier) & MVC.* [Online]. 2013.
URL: http://queens.db.toronto.edu/~papaggel/courses/csc309/docs/
lectures/Web-architectures.pdf.
Accessed 25 March 2013.

PennState, College of Earth and Mineral Sciences: *Evolotion of Web Mapping Technology.* [Online]. 2009.
URL: https://www.e-education.psu.edu/geog863/resources/l3_p3.html.
Accessed 24 March 2013.

PostgreSQL Global Development Group: *PostgreSQL Documentation.* [Online] 2013.
URL: http://www.postgresql.org/docs/.
Accessed 8 May 2013.

QGIS Org.: *QGIS.* [Online].
URL: http://www.qgis.org/en/site/.
Accessed 27 October 2013.

R.K.: *Geographic Information Systems.* [Online]. 2012.
URL: http://gis.stackexchange.com/
questions/18838/80-of-data-has-a-spatial-component-says-who.
Accessed 13 March 2013.

Rankin, B.: *Projection Reference.* [Online]. 2006.
URL: http://www.radicalcartography.net/?projectionref.
Accessed 18 April 2013.

Rouse, M.: *SearchBusinessAnalytics - heat map (heat map or tree map).*
[Online]. 2011.
URL: http://searchbusinessanalytics.techtarget.com/definition/heat-map.
Accessed 20 April 2013.

Salmi, T.: *Teknologiaselvitys - Paikkatietojärjestelmien teknologiavaihtoehdot.* 2012.
Viestintävirasto, Helsinki: Karttakeskus.

Schut, P.: *OpenGIS® Web Processing Service.* 1.0.0. 2007.
Open Geospatial Consortium Inc.

Skau, D.:*How to Use Maps in Data Visualization.* [Online]. 2012.
URL: http://blog.visual.ly/you-are-here-using-maps-in-data-visualization/.
Accessed 20 April 2013.

Spatial Information Clearinghouse: *Functions of a Geographic Information System.*
[Online]. 2004.
URL: http://maic.jmu.edu/sic/gis/functions.htm.
Accessed 17 March 2013.

Spatial Reference Org.: *Spatial Reference.* [Online]. 2013.
URL: http://spatialreference.org/.
Accessed 1 Appril 2013.

SQLite: *SQLite.* [Online]. 2013.
URL: http://www.sqlite.org/.
Accessed 6 April 2013.

Tikunov, V. S.: *An Alternative View of The World.* [Art] (UNEP). 2002.

TopoGrafix: *GPX The GPS Exchange Format.* [Online]. 2013.
URL: http://www.topografix.com/gpx.asp.
Accessed 15 February 2013.

Twitter Inc.: *Twitter.* [Online]. 2013.
URL: https://twitter.com/.
Accessed 6 April 2013.

University of Colorado: *Elements Tthat Are Found on Virtually All Maps.* [Online]. 1996.
URL: http://www.colorado.edu/geography/gcraft/notes/cartocom/elements.html.
Accessed 19 April 2013.

USGS: *USGS.* [Online]. 2007.
URL: http://egsc.usgs.gov/isb/pubs/gis_poster/.
Accessed 9 March 2013.

W3C: *Web Service Tutorial.* [Online]. 1999 - 2013.
URL: http://www.w3schools.com/Webservices/.
Accessed 26 March 2013.

W3C: *W3C Standards.* [Online]. 2012.
URL: www.w3c.org/standards/.
Accessed 25 March 2013.

Ward, M., Grinstein, G. & Keim, D.: *Interactive Data Visualization - Foundations, Techniques, and Applications.* 2010.
Natic: A K Peters, Ltd.

Watkins, C.:*Vector vs. Raster.* [Online]. 2004-2010.
URL: http://www.illustratortips.com/index.php/Instruction/
Beginner-Tips/vector-vs-raster.html.
Accessed 14 April 2013.

Wikimapia. *Wikimapia.* [Online]. 2013.
URL: www.wikimapia.org.
Accessed 6 April 2013.

Wikipedia Org: *Helsinki Tram.* [Online]. 2013.
URL: http://en.wikipedia.org/wiki/Helsinki_tram.
Accessed 20 April 2013.

Wikipedia Org: *Graphical User Interface.* [Online]. 2013.
URL: http://en.wikipedia.org/wiki/Graphical_user_interface.
Accessed 13 April 2013.

Wikipedia Org, *List of map projections.* [Online]. 2013.
URL: http://en.wikipedia.org/wiki/List_of_map_projections.
Accessed 18 April 2013.

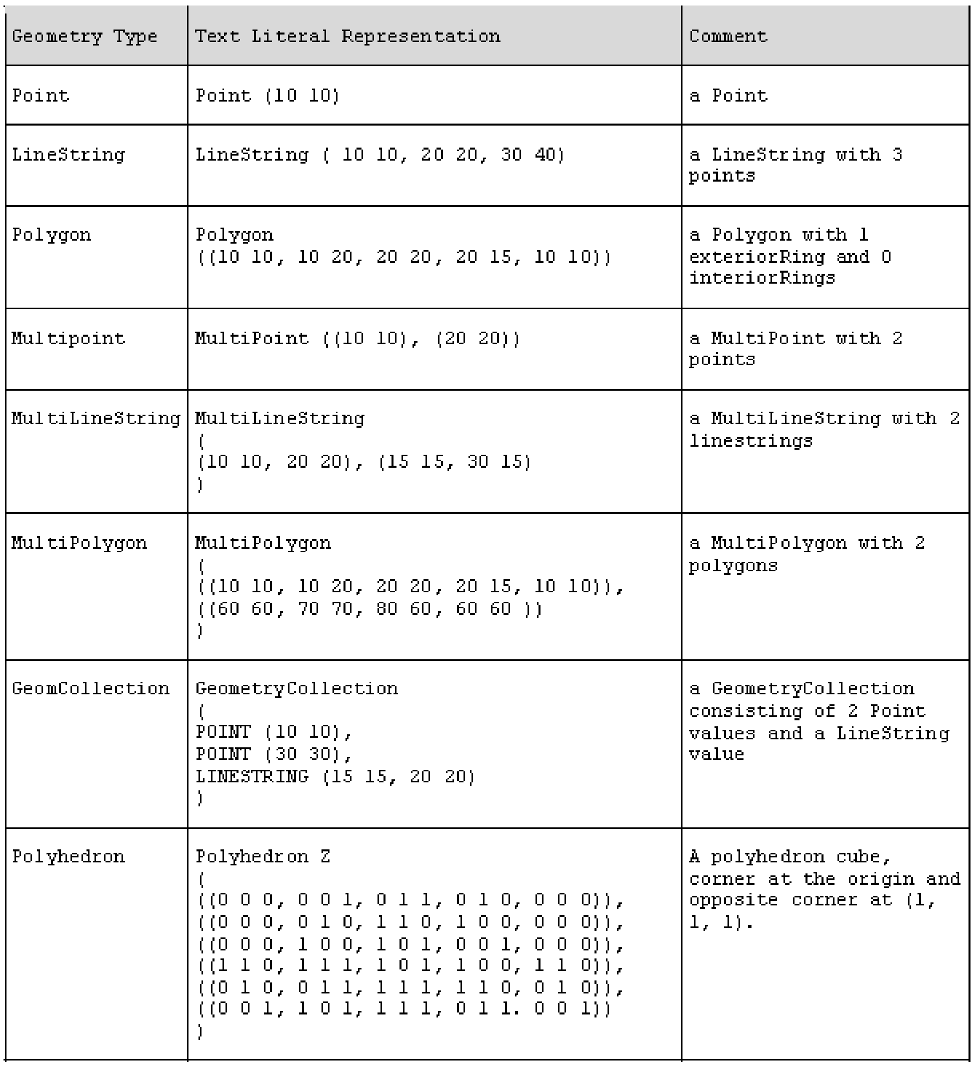
Wright, D. J., Goodchild, M. F. & Proctor, J. D.: Demystifying the Persistent Ambiguity of GIS as "Tool" Versus "Science". *The Annals of the Association of American Geographers,* 87(2), pp. 346-362. 1997.

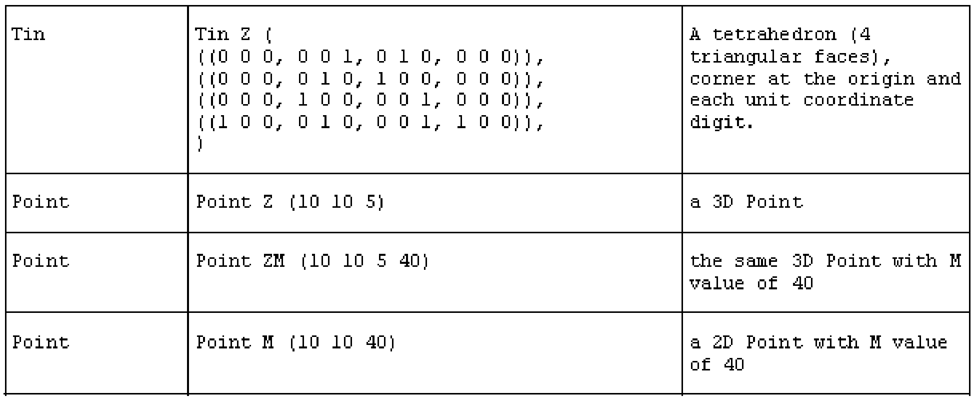
Yahoo Inc.: *Flickr.* [Online]. 2013.
URL: http://www.flickr.com/.
Accessed 6 April 2013.

Yau, N.: *Visualize This, The FlowingData Guide to Design, Visualization, and Statistics.* 1st ed. 2011.
Indianapolis: Wiley Publishing, Inc.

Appendix 1. Well-known Text Representation of Geometry Data

Table of Geometry Types and Their Text Literal Representations – Copyright (C) 2010 Open Geospatial Consortium, Inc. – From source: OGC specification 06-103r4, p 63-64.

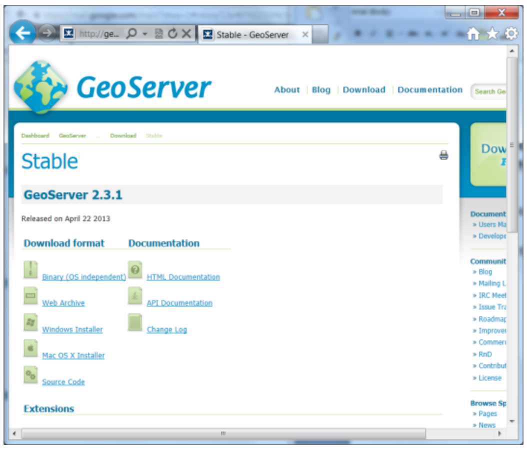




Geometry Type	Text Literal Representation	Comment
Point	Point (10 10)	a Point
LineString	LineString (10 10, 20 20, 30 40)	a LineString with 3 points
Polygon	Polygon ((10 10, 10 20, 20 20, 20 15, 10 10))	a Polygon with 1 exteriorRing and 0 interiorRings
Multipoint	MultiPoint ((10 10), (20 20))	a MultiPoint with 2 points
MultiLineString	MultiLineString ((10 10, 20 20), (15 15, 30 15))	a MultiLineString with 2 linestrings
MultiPolygon	MultiPolygon (((10 10, 10 20, 20 20, 20 15, 10 10)), ((60 60, 70 70, 80 60, 60 60)))	a MultiPolygon with 2 polygons
GeomCollection	GeometryCollection (POINT (10 10), POINT (30 30), LINESTRING (15 15, 20 20))	a GeometryCollection consisting of 2 Point values and a LineString value
Polyhedron	Polyhedron Z (((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)), ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)), ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1. 0 0 1)))	A polyhedron cube, corner at the origin and opposite corner at (1, 1, 1).
Tin	Tin Z (((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 0 0 1, 0 0 0)), ((1 0 0, 0 1 0, 0 0 1, 1 0 0)),)	A tetrahedron (4 triangular faces), corner at the origin and each unit coordinate digit.
Point	Point Z (10 10 5)	a 3D Point
Point	Point ZM (10 10 5 40)	the same 3D Point with M value of 40
Point	Point M (10 10 40)	a 2D Point with M value of 40

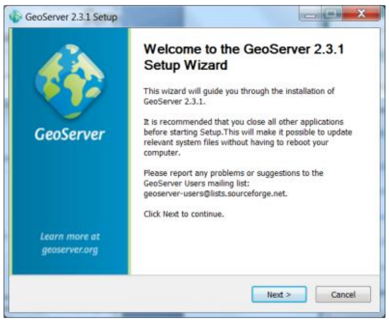
Appendix 2. GeoServer Setup for Windows

1. Download the GeoServer from http://geoserver.org/display/GEOS/Stable



2. Select the Windows Installer and download GeoServer package (geoserver-2.3.1.exe)

3. Execute the installer



Welcome to the GeoServer 2.3.1 Setup Wizard

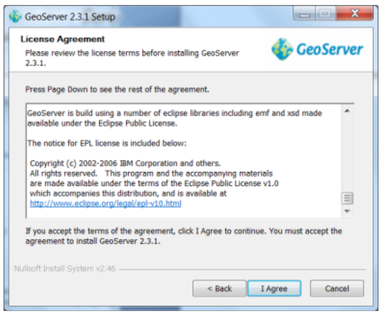
This wizard will guide you through the installation of GeoServer 2.3.1.

It is recommended that you close all other applications before starting Setup.This will make it possible to update relevant system files without having to reboot your computer.

Please report any problems or suggestions to the GeoServer Users mailing list: geoserver-users@lists.sourceforge.net.

Click Next to continue.

4. Press next



License Agreement

Please review the license terms before installing GeoServer 2.3.1.

Press Page Down to see the rest of the agreement.

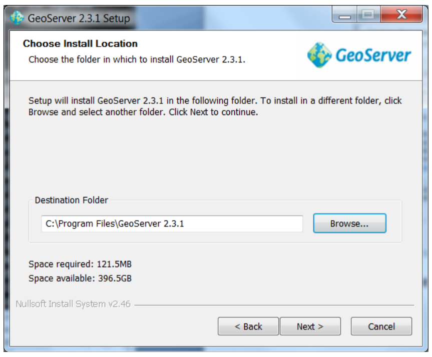
GeoServer is build using a number of eclipse libraries including emf and xsd made available under the Eclipse Public License.

The notice for EPL license is included below:

Copyright (c) 2002-2006 IBM Corporation and others.
All rights reserved. This program and the accompanying materials
are made available under the terms of the Eclipse Public License v1.0
which accompanies this distribution, and is available at
http://www.eclipse.org/legal/epl-v10.html

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install GeoServer 2.3.1.

5. Agree the license



Choose Install Location
Choose the folder in which to install GeoServer 2.3.1.

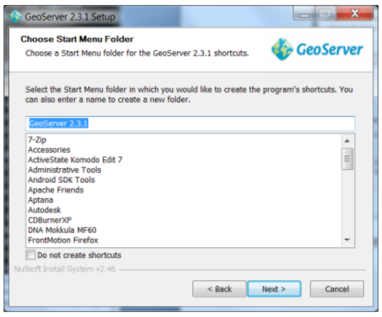
Setup will install GeoServer 2.3.1 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder
C:\Program Files\GeoServer 2.3.1

Space required: 121.5MB
Space available: 396.5GB

Nullsoft Install System v2.46

6. Choose the Install location and press Next>



Choose Start Menu Folder
Choose a Start Menu folder for the GeoServer 2.3.1 shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

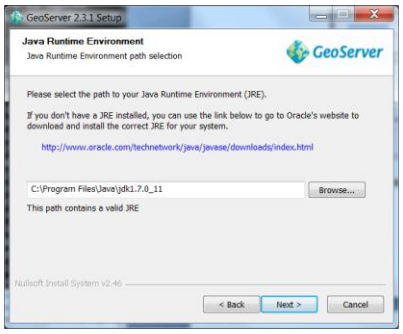
GeoServer 2.3.1

7-Zip
Accessories
ActiveState Komodo Edit 7
Administrative Tools
Android SDK Tools
Apache Friends
Aptana
Autodesk
CDBurnerXP
DNA Mokkula MF60
FrontMotion Firefox

☐ Do not create shortcuts

Nullsoft Install System v2.46

7. Select the Start Menu folder and press Next>



Java Runtime Environment
Java Runtime Environment path selection

Please select the path to your Java Runtime Environment (JRE).

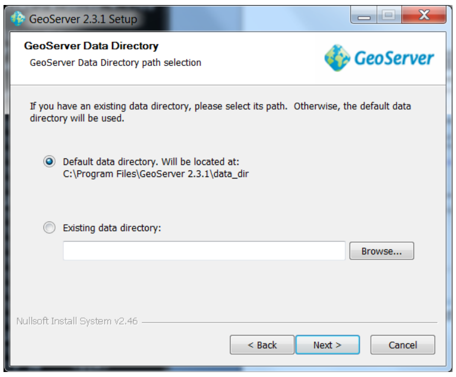
If you don't have a JRE installed, you can use the link below to go to Oracle's website to download and install the correct JRE for your system.

http://www.oracle.com/technetwork/java/javase/downloads/index.html

C:\Program Files\Java\jdk1.7.0_11
This path contains a valid JRE

Nullsoft Install System v2.46

8. Select the path where your Java JDK is installed and press Next>



GeoServer Data Directory
GeoServer Data Directory path selection

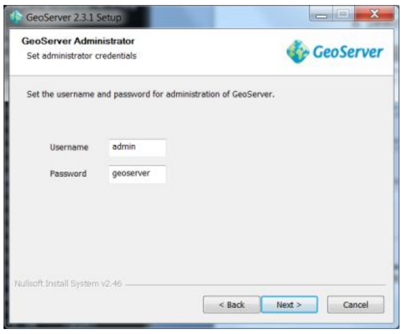
If you have an existing data directory, please select its path. Otherwise, the default data directory will be used.

● Default data directory. Will be located at:
C:\Program Files\GeoServer 2.3.1\data_dir

○ Existing data directory:

Nullsoft Install System v2.46

9. Press Next> unless you want to change the data directory for GeoServer



GeoServer Administrator

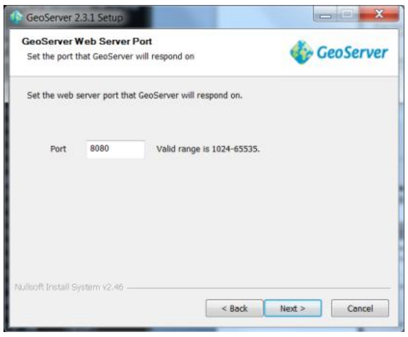
Set administrator credentials

Set the username and password for administration of GeoServer.

Username admin

Password geoserver

10. Set the username and password as instructed and press Next>



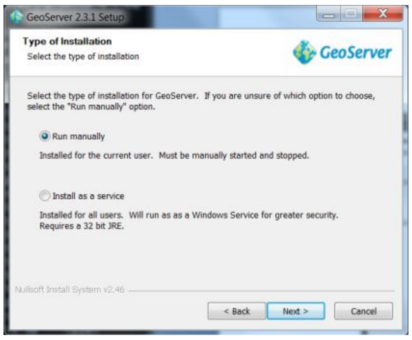
GeoServer Web Server Port

Set the port that GeoServer will respond on

Set the web server port that GeoServer will respond on.

Port 8080 Valid range is 1024-65535.

11. Set the TCP/IP port that GeoServer will respond.



Type of Installation

Select the type of installation

Select the type of installation for GeoServer. If you are unsure of which option to choose, select the "Run manually" option.

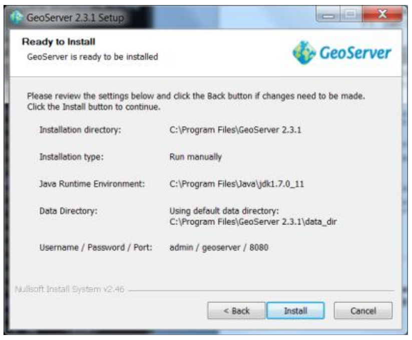
[✓ Run manually]

Installed for the current user. Must be manually started and stopped.

Install as a service

Installed for all users. Will run as as a Windows Service for greater security. Requires a 32 bit JRE.

12. Select how you want your GeoServer is Run, press Next>



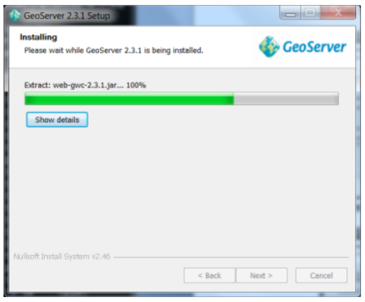
Ready to Install

GeoServer is ready to be installed

Please review the settings below and click the Back button if changes need to be made. Click the Install button to continue.

Installation directory:	C:\Program Files\GeoServer 2.3.1
Installation type:	Run manually
Java Runtime Environment:	C:\Program Files\Java\jdk1.7.0_11
Data Directory:	Using default data directory: C:\Program Files\GeoServer 2.3.1\data_dir
Username / Password / Port:	admin / geoserver / 8080

13. Press Next>

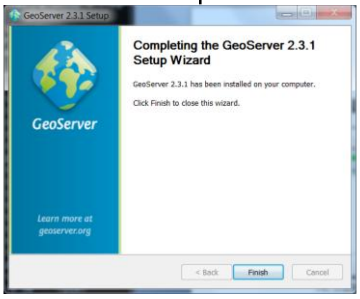


Installing

Please wait while GeoServer 2.3.1 is being installed.

Extract: web-gwc-2.3.1.jar... 100%

14. Installation process will begin.

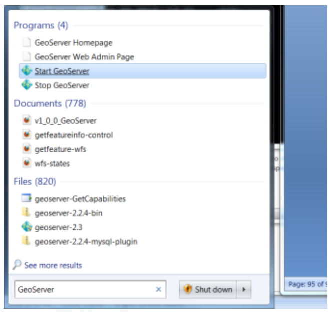


Completing the GeoServer 2.3.1 Setup Wizard

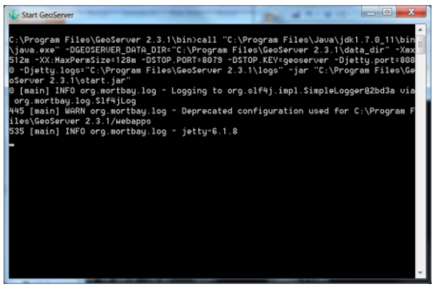
GeoServer 2.3.1 has been installed on your computer.

Click Finish to close this wizard.

15. Press Finish.

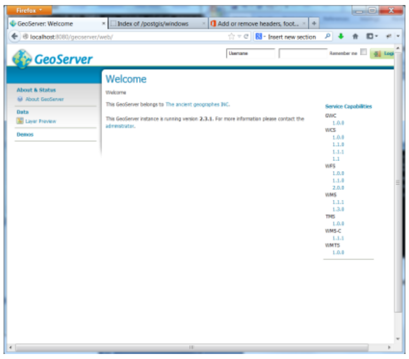


16. Start GeoServer and this will launch the GeoServer



```
C:\Program Files\GeoServer 2.3.1\bin>call "C:\Program Files\Java\jdk1.7.0_11\bin
\java.exe" -DGEOSERVER_DATA_DIR="C:\Program Files\GeoServer 2.3.1\data_dir" -Xmx
512m -XX:MaxPermSize=128m -DSTOP.PORT=8079 -DSTOP.KEY=geoserver -Djetty.port=808
0 -Djetty.logs="C:\Program Files\GeoServer 2.3.1\logs" -jar "C:\Program Files\Ge
oServer 2.3.1\start.jar"
0 [main] INFO org.mortbay.log - Logging to org.slf4j.impl.SimpleLogger@2bd3a via
 org.mortbay.log.Slf4jLog
445 [main] WARN org.mortbay.log - Deprecated configuration used for C:\Program F
iles\GeoServer 2.3.1/webapps
535 [main] INFO org.mortbay.log - jetty-6.1.8
_
```

17. Open Web browser and open the admin page http://localhost:8080/geoserver/Web/



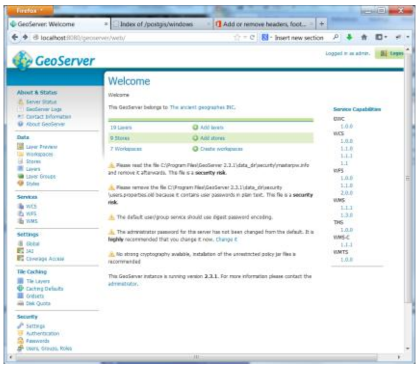
Welcome

Welcome

This GeoServer belongs to The ancient geographes INC.

This GeoServer instance is running version 2.3.1. For more information please contact the administrator.

18. Login using the username and password you set in install phase.



Welcome

Welcome

This GeoServer belongs to The ancient geographes INC.

19 Layers	Add layers
9 Stores	Add stores
7 Workspaces	Create workspaces

Please read the file C:\Program Files\GeoServer 2.3.1\data_dir\security\masterpw.info and remove it afterwards. The file is a **security risk.**

Please remove the file C:\Program Files\GeoServer 2.3.1\data_dir\security\users.properties.old because it contains user passwords in plain text. This file is a **security risk.**

The default user/group service should use digest password encoding.

The administrator password for this server has not been changed from the default. It is **highly** recommended that you change it now. Change it

No strong cryptography available, installation of the unrestricted policy jar files is recommended

This GeoServer instance is running version **2.3.1**. For more information please contact the administrator.

19. And now you are ready to play with GeoServer.

20. The GeoServer can be stopped by running the Stop GeoServer same way as it was started (or just closing the window running GeoServer)

Appendix 3. GeoServer Configuration Parameters: The Biking Route

GeoServer version number: 2.3.1

DATABASE AND TABLE PARAMETERS

Database brand: PostgresSQL with PostGIS add-on

Database Connection paramaters:

host: `localhost` (default)

port: `5432` (default)

Database name: `postgis`

Database schema: `public`

Database user: `postgres`

Data table name: `biking_route`

Data table geometry columns name: `geom`

Data table geometry columns SRID: `4326` (WGS84)

WORKSPACES – EDIT WORKSPACE - PARAMETERS

Name: `Master_Thesis`

Namespace URI: `master_thesis`

Other parameters as set by default

STORES – EDIT VECTOR DATA STORE (PostGIS) - PARAMETERS

Basic Store Info

Workspace: `Master_Thesis`

Data Source Name: `Master`

Description: `Use case test – Biking Route Data`

Enabled: *V*

Connection Parameters

host: `localhost`

port: `5432`

database: `postgis`

schema: `public`

user: `postgres`

password: `*****`

Other parameters as set by default

WORKSPACES – EDIT WORKSPACE - PARAMETERS

Name: `Master_Thesis`

Namespace URI: `master_thesis`

Other parameters as set by default

LAYERS – NEW LAYER - PARAMETERS

Add layer from: `Master_Thesis:Master`

Select Layer name: `biking_route` Click Publish

LAYERS – EDIT LAYER - PARAMETERS

Master_Thesis:biking_route

DATA Tab

Basic Resource Info

Name: `biking_route`

Title: `biking_route`

Abstract: `free form of text`

Keywords

Current Keywords: `as set by default.`

Coordinate Reference Systems

Native SRS: `EPSG:4326`

Declared SRS: `EPSG:4326`

SRS Handling: `Force declared`

Bounding Boxes

Native Bounding Box: `Compute from data`

Lat/Lon Bounding Box: `Compute from data`

Other parameters as set by default including the Publish, Dimension and Tile Caching tabs.
