Student thesis series INES nr 224

Developing Android Mobile Map Application with Standard Navigation Tools for Pedestrians

Eduard Mikayelyan

2011
Department of
Physical Geography and Ecosystems Science
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden

Eduard Mikayelyan (2011). *Developing Android Mobile Map Application with Standard Navigation Tools for Pedestrians*
Master degree thesis, 30 credits in Geomatics
Department of Physical Geography and Ecosystems Science, Lund University

Level: Master of Science (MSc)

Course duration: September 2010 until January 2011

Disclaimer

Developing Android Mobile Map Application with Standard Navigation Tools for Pedestrians

---

Eduard Mikayelyan

Master thesis, 30 credits, in Geomatics

Supervisor Lars Harrie
Division of Physical Geography and Ecosystems Analysis
Department of Earth and Ecosystem Sciences
Lund University

Abstract

The mobility of mobile devices has made it possible to develop and use maps and map based applications for navigation purposes. Since most mobile map applications nowadays are developed for motor vehicles, there is a demand for portable pedestrian navigation applications. In this thesis the Android mobile map application with standard navigation tools for pedestrian navigation was developed, as a platform for facilitating the Lund Challenge location based demonstrator of the HaptiMap project. The aim of the demonstrator is to make the sights of Lund city more accessible. The mobile phone application is being designed as a touristic, historical location based game which will also assist tourists to navigate themselves in the city. To enable exploration of historical and current sites of Lund the demonstrator should contain basic components of exploring and way finding. Prior to the development the OpenStreetMap (OSM) road network data and Swedish National Road Database (NVDB) were introduced. The main advantage of using the OSM data over the NVDB dataset is the completeness of the OSM data in terms of pedestrian paths. The datasets were imported to PostgreSQL spatially extended PostGIS database, where different routing algorithms provided by pgRouting were used for routing calculations. As the Lund Challenge demonstrator is intended not only for general users but also for visually impaired users, the problem of user navigation in the parks and open areas were also discussed and the feasibility study was performed. The limitation of the developed application was the problem of the user navigation in the parks and open areas. It is therefore necessary to upgrade the road database with possible path in the open areas and parks in order to implement this application.

Keywords: Geographic Information System (GIS), Pedestrian navigation, Android, HaptiMap, Lund Challenge, OpenStreetMap (OSM), Swedish National Road Database (NVDB), PostGIS, Navigation in parks and open areas.

Table of Content

List of Abbreviations

ADT	-Abstract Data Type
ADT[1]	-Android Development Tools
API	-Application Programming Interface
CAD	-Computer Aided Design
CEN	-the European Committee for Standardization
CityGML	-City Geography Markup Language
DB	-Database
DBMS	-Database Management System
DDL	-Data Definition (or Description) Language
DIS	-Distributed Interactive Simulation
DML	-Data Manipulation Language
DTD	-Document Type Definition
DTM	-Digital Terrain Model
ERDB	-Extensible Relational Database
ESRI	-Ergonomics and Safety Research Institute (UK)
HTTP	-Hypertext Transfer Protocol
ISO	-International Organization for Standardization
GeoJSON	-Geospatial data interchange format based on JavaScript Object Notation
GIS	-Geographic Information System
GML	-Geography Markup Language
GPL	-General Public License
GPS	-Global Positioning System
KML	-Keyhole Markup Language
LOD	-Level of Details
MPEG-4	-Motion Picture Export Group - 4
NVDB	-Nationella Vägdatabasen
OGC	-Open Geospatial Consortium
OODB	-Object-Oriented Database
OS	-Operation System
OSM	-Open Street Map
PDA	-Personal Digital Assistant
PND	-Personal Navigation Device
POI	-Point of Interest
RDB	-Relational Database
SDK	-Software Development Kit
SVG	-Scalable Vector Graphics
SQL	-Structured Query Language
UI	-User Interface
URL	-Uniform Resource Locator
VGI	-Volunteered Geographic Information
VRML	-Virtual Reality Modeling Language
WebCGM	-Web Computer Graphics Metafile
TMS	-Tile Map Service
WMS	-Web Map Service
WPS	-Web Processing Service
XLink	-XML Linking Language
XML	-Extensible Markup Language

CHAPTER 1. Introduction

1.1. Background

Maps and route services are vital for navigation. For instance, the personal and interactive use of maps in mobile environment has several advantages. The mobility of mobile devices makes it possible to use maps for navigational purposes. On the other hand in desktop applications maps are used mainly for route planning. The rapid development of new technologies enables high speed internet in mobile devices which further makes it possible to use advanced online services for navigation. Most mobile map applications nowadays are developed for cars; there is a lack of services for pedestrian and cycle navigation.

The HaptiMap project aims at developing pedestrian navigation systems. The motivation of this thesis is to complement the HaptiMap project in developing the location based demonstrator that is being developed on Android mobile operation system platform. Lund University and Lund Municipality are cooperating in the HaptiMap project to develop the "Lund Challenge" demonstrator. The demonstrator is intended to make the sights of Lund city more accessible from mobile phone application which will help tourists to navigate themselves in Lund city. It should be designed as a touristic, historical location based game. The application should not be functional only for general users but it also should function for visually impaired users. The demonstrator should contain basic components of exploring and way finding including the location of historical sites which would make exploring them possible (HaptiMap, 2010).

Prior to Android application development the study of two datasets used in the HaptiMap project was performed. Additionally the OpenStreetMap (OSM) and Swedish National Road Database (NVDB) road network data were imported to PostGIS database and the routing calculation were performed using the pgRouting functionality.

OSM is an example of volunteered geographic information. It is an open source project that allows adding, editing and retrieving geographic data. The *Google Mobile Maps* application provides exploring tools for navigating in the city. Similar application is necessary to develop based on OSM data. While in the contrast to Google Mobile Maps the OSM data and the navigation tools based on it include pedestrian paths for routing that have the largest potential demand. Furthermore, since the development of OSM data is an ongoing process and it is open source project a lot more pedestrian paths data are expected that will assist better pedestrian route finding.

The Lund Challenge demonstrator should include standard navigation tools for navigating users in the city. Miguel Molina is developing it at Faculty of Engineering in Lund University. The demonstrator should include voice assistance which will allow visual impaired users to get information about the sights and navigate themselves in the city. The purpose of this study is to

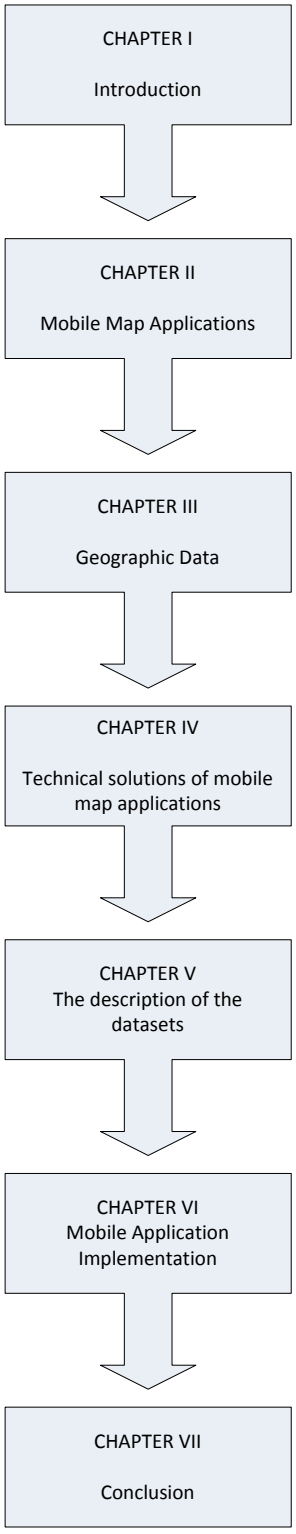
develop Android mobile map application with standard navigation tools for pedestrians. The latter, as a prototype, would be used in future developments of the Lund Challenge demonstrator.

1.2. Aim

The pivotal aim of this master thesis is to complement the tourist oriented “Lund Challenge” Android mobile application by adding some functionality in already developed toolkit. For this purpose an Android application would be developed and as a prototype would be used to supplementing the routing functionality and simple navigation tools to the “Lund Challenge” demonstrator.

1.3. Disposition of the Thesis

The structure of the chapters in the thesis is as follows:



This chapter discusses the background, introduces the problems and formulates the foundations why the problem is of interest.

In this chapter the Mobile Map applications are introduced to give a background of the followed work. Also the HaptiMap project is described in this chapter.

This chapter presents the theoretical background of geographical data and geographic data quality.

In this chapter the intentions of this study was introduced. The chapter starts by discussing the background of geographic data storage in different file formats and databases, furthermore presenting the theoretical background of the client-server architecture and web services and ends with an overview of the Android mobile operation system.

In this chapter the description of OpenStreetMap (OSM) and Swedish National Road Database (NVDB) datasets are given.

This chapter presents the implementation and discussion of the developed Android application.

This chapter concludes and presents the outcomes of the Android application development.

CHAPTER 2. Mobile Map Applications

Nowadays maps are used more for personal and interactive purposes than for just visualization of spatial data. Due to technological progress it is now possible to use web maps in mobile device environment. By use of wireless internet from geospatial services, map data can be distributed to mobile devices like mobile phones or Personal Digital Assistant (PDA). Because maps on mobile devices are used also in portable situations, they should be presented in totally different way than on static personal desktops. Similar to screen-based maps mobile maps also can be explored, updated and analyzed interactively (Meng et al., 2005). While most current mobile map applications are made for car navigation, applications for cyclists and pedestrian navigation also exist (Nivala, 2005).

2.1. Pedestrian Navigation in Stockholm

A good example of pedestrian navigation application is the "Pedestrian Navigation in Stockholm" described in Dawidson (2009). Thus the aim of this project is to develop mobile phone and PDA based application to enable visually impaired pedestrians to navigate and explore unfamiliar territories (areas) within the urban environment. Aside from it the project was designed to aid pedestrians and visually impaired users in outdoor navigation (including public transportation), it was also intended to encompass indoor navigation (Dawidson, 2009).

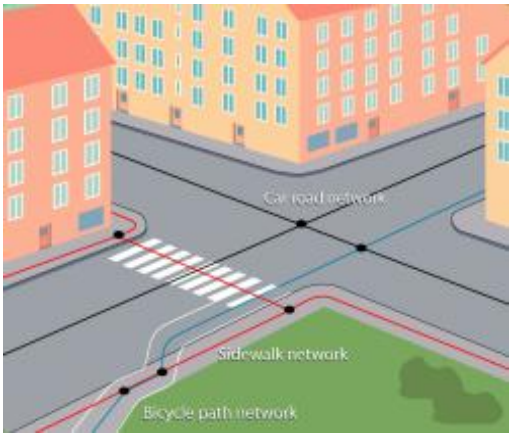


Figure 2.1. Pedestrian network with associated links to cycle and car roads (e-Adept, 2010).

To achieve this goal digitized pedestrian and cyclist path networks and advance positioning techniques such as indoor positioning were used. It is worth noting, road network data, which include pedestrian network as an attribute, is not sufficient for the purpose of this project, therefore pedestrian network was used. Pedestrian network includes own pedestrian nodes and links to cycle paths and roads (Figure 2.1). The pedestrian database includes a huge amount of information like addresses, bus stops, street lighting, points of interests, stairs and also

information about barriers like road constructions and bike stands. The database is being continuously updated to have up to date information for real time navigation (Dawidson, 2009).

2.2. Google Maps for Mobile

An example of a navigation style is the free mobile phone Java application called “Google Maps for Mobile” launched by Google. The application includes GPS (Global Positioning System) like location service which makes possible to identify the location of the user without using GPS receivers, where the location information is derived through the nearest wireless networks and cell sites. The software is used to search for the known wireless networks and cell sites databases to localize the user and also technique of signal strength triangulation from cell antennas is utilized where the location of the antennas are used to supplement the location discovery. The order of services utilized for location finding in the application is as follows: GPS-based services, WLAN- or WiFi- based services and at last the cell triangulation-based services (Google Maps for Mobile, 2011b).

Besides the enhanced location finding services the application includes various services supporting car and pedestrian navigation such as *Navigation (Beta), Places with Hotpot, 3D Maps, Compass Mode, Offline Reliability, Latitude, Street View, Traffic*, etc. (Google Maps for Mobile, 2011a). The Navigation service is providing turn-by-turn GPS navigation service combined with voice assistance for both cars and pedestrians. The Place with Hotpot service is provides an easy method of search for POIs (Point of Interest) and even provides the personalized advices from Google. The 3D Maps service provides 3D map view functionality, with enhanced twist with two finger functionality. The Compass Mode service is rotating the maps to direct the user. The Latitude service is supplying the application with location sharing functionality which helps to find the friends and allow friends to find own position. The Street View service provides the service for enhancing the place finding by providing street-level images and also helps the user to find one's bearings on the ground. The Traffic service provides online traffic situation and aids to find the fastest route (Google Maps for Mobile, 2011a).

All these services collectively are providing advance navigation functionality to the user and as it was mentioned Offline Reliability and further Offline Rerouting are assisting the user in the case of connection failure; however the Internet connection is required to exhaust all the services of the application (Google Maps for Mobile, 2011a).

The most of these services are compatible with most mobile platforms, though some features are available only for certain platforms (Table 2.1) (Google Maps for Mobile, 2011a).

Table 2.1. Supported features of “Google Maps for Mobile” application by different mobile platforms (Google Maps for Mobile, 2011a).

Feature	Android	BlackBerry	iPhone	Nokia or S60	Windows
Navigation (Free, turn-by-turn GPS system)	✓				
Places with Hotpot	✓				
3D Maps	✓				
Compass mode	✓		✓		
Offline reliability	✓				
Place pages (Details, reviews, and more)	✓	✓			
Business listings (Address, phone numbers, etc.)	✓	✓	✓	✓	✓
My Location	✓	✓	✓	✓	✓
Latitude	✓	✓	✓ App Store	✓	✓
Voice Search	✓	✓		✓	✓
Traffic	✓	✓	✓	✓	✓
Street View	✓	✓	✓	✓	✓
Driving directions	✓	✓	✓	✓	✓
Transit directions	✓	✓	✓	✓	✓
Biking directions	✓	✓			
Walking directions	✓	✓	✓	✓	✓
Layers	✓	✓		✓	✓
Satellite layer	✓	✓	✓	✓	✓
Terrain layer	✓				
My Maps	✓	✓		✓	✓
Starred Items	✓	✓		✓	✓
Labs	✓	✓			

2.3. HaptiMap project

The HaptiMap project aims to create maps and to develop location-based services for all users including visually impaired and elderly people. Also the goal of the project is to develop tools for simplifying the development of the applications by adapting already developed components. The HaptiMap project has 13 partners including the Lund University and Lund Kommun (Haptimap Homepage, 2011). The demonstrators to be created of the HaptiMap project are (cf. Haptimap, 2010):

1. PocketNavigator
2. NAVTEQ Connect
3. Lund Challenge
4. Event Guide - JUICY BEATS
5. Tour Guide - TANDEM
6. Exhibition Guide – EMSCHERKUNST
7. The Terrain Navigator
8. Content profiler
9. Developer-targeted demonstrators

The project covers the application scenarios such as exploration, navigation, route planning in the city and countryside environment, games, etc. (HaptiMap, 2010).

CHAPTER 3. Geographic Data

The definition of geographic data from ISO (International Organization for Standardization) standards is (ISO41): “Data with implicit or explicit reference to a location relative to the Earth”. Geographic data are much more than digital maps. They are spatially referenced data describing objects in space, which includes geometric and thematic data. The geographic data can be used in geographic information system (GIS) for variety of scientific analysis and map production. An example of an application uses geographic data is pedestrian navigation and for this application data includes not only geometry objects like points, lines and polygons, but also more complex geometric and thematic properties like connectivity, path types, etc.

3.1. Geographic Data Types

Two data structures exist for geographic data representation: vector structure and raster structure. These two data structures were developed for different applications.

3.1.1. Vector structure

In vector structure geographic features are represented as geometric shapes. The basic geometric shapes used in vector structure are point, line (polyline) and polygon (Figure 3.1). Point is a 0-dimensional geometric object which is fixed by *x* (or latitude), *y* (or longitude) and optional *z* (or altitude) coordinates and is representing certain location in geographic space. For example, bus stops can be represented as points in the city scale.

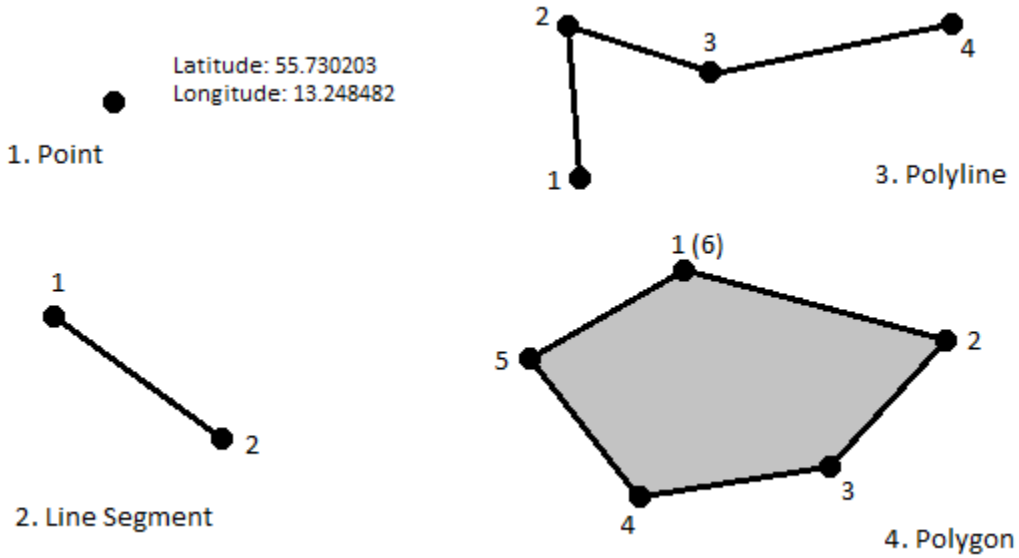


Figure 3.1. Simple geometric objects.

The line segment is a simple type of the line and more complex type is a polyline. Line segment is a 1-dimensional geometric object and is represented as a straight line between two points: the start point and the end point. The line segments attached to each other in one's start points to another's end points is called polyline. The distance can be calculated from the line object. Nodes are the start and end points for the line and for polylines, while a point in-between the start and the end point is called a "vertex". Polylines can be used to represent the roads for example.

The polygon is a 2-dimensional geometric object and is represented as a circumscribed area by the polyline, whose start and end points are matching. The area and perimeter can be calculated from the polygon object. The representation of certain geographic features as a polygon or point is based on the scale, purpose, etc.

In vector structure geographic objects are linked to attribute table by use of object identifier. There are different models of storing vector data. The simplest model of storing vector data is called *spaghetti model*. In spaghetti model the geometric objects: points, lines and polygons, are stored as a single or many pairs of coordinates in consecutive order and just for polygons first node and the last node should match otherwise it would not be a polygon but just a polyline. More complex models for storing vector data is called *topological models*.

Topological properties of the geographic objects are properties that are invariants in topological transformations. To understand the meaning of a topological transformation it is enough to consider the Euclidean plane as a sheet of rubber which is possible to stretch and compress to any degree without tearing it. In this condition some properties of the object on the plane will remain, for example two polygons sharing the same edge before the transformation will maintain this property after any kind of topological transformation; however, the properties like the perimeter or area of the polygons will change. The topological model consists of various branches; most common branches are point-set topology and combinatorial (algebraic) topology Worboys and Duckham, 2004).

In topological model, the vector data that is stored contains not only the information of the geometric objects as a coordinates but also the information about neighboring objects is stored jointly with the coordinates.

3.1.2. Raster structure

In the raster structure the data is stored in cells in matrices, where each cell is assigned a number which is corresponding to a certain attribute. As raster structure stores only one number for each cell it is possible to extend it to represent in RGB (red-green-blue) colors by using raster bands.

Even though it is possible to represent discrete objects in raster structure; however, the raster structure is most commonly used for representing continuous surfaces like topography, precipitation, etc. The raster structure will not be described in details as it is not used in this study.

3.2. Geographic data quality

The quality of geographic data by ISO standards is defined as "Totality of characteristics of a product that bare on its ability to satisfy stated and implied needs" (ISO34 2002). The quality of geographic data may render it valuable or invaluable in any analysis. Different GIS applications acquire geographic datasets of different rank of quality. With the rapid increase in sharing geographic data, the data are often used in other application than they were originally intended, and the quality of information is important because it determines the choice of geographic data to satisfy application requirements (Kresse and Fadaie, 2004).

The quality standards for geographic data are divided into data quality elements and data quality overview elements. The data quality elements include five subgroups defined by ISO 19113, which includes the quantitative quality information (Kresse and Fadaie, 2004):

- *Completeness* defines the fact if data includes features, attributes and relationships. An unfavorable example can be the absence of turn restrictions in road network for car navigation application.
- *Logical consistency* defines the fact if data remains the logical rules defined in data model. Example: The data model intended that data should contain full topological model, but it contains only Spaghetti model.
- *Positional accuracy* is the features positional accuracy.
- *Temporal accuracy* is the accuracy of temporal attributes and relationships of the features. For example: The date of data creation is 10 October 2010.
- *Thematic accuracy* includes correctness of quantitative and non-quantitative attributes, and also correctness of features classification and relationships. For example: In road network the path classified as a cycle road, which is pedestrian path in reality.

The data quality overview elements include non-quantitative quality information and consist of three subgroups defined by ISO 19113 (Kresse and Fadaie, 2004):

- *Purpose*: It is the description of intentional use of the data and the motivation of its creation.
- *Usage*: It illustrates how the data was used in application and what extent.
- *Lineage*: It is the description of data history including the creation process and other processes that data went trough.

3.3. Project data

There are a great number of geographic datasets. In this study we use two of the datasets that are used in the HaptiMap project: OpenStreetMap (OSM) data and NVDB (Nationella Vägdatabases) / Lund Municipality data.

3.3.1. OpenStreetMap

OSM is an open source map service which creates and provides free geographic data. OSM is an example of volunteered geographic information VGI. People collect geographic data from several sources like a GPS devices and open source satellite imagery and later on upload that information to OSM's database; in this way the user can update, add or correct the map data in that area.

"Open source is a free sharing system that can be contributed by everyone who wants to work on it and can benefit everyone who wants to use it" (Jia, 2010, p. 5). Open source does not regards only geographic information or computer community; it can be widen to various fields for example sharing the cooking recipes and so on.

To understand the creditability of the digital open source content we can compare with the website of the Wikipedia. Wikipedia is using web 2.0 technology or so called wiki technology. In web 1.0 technology users could only obtain or retrieve information from the websites, however web 2.0 technology is bidirectional, thus it also makes possible to upload the data, download data or edit it online. Using the open edit model everyone can contribute Wikipedia by writing or updating the articles. The number of Wikipedia articles reaches to approximately 3.6 million in 2011. Some issues are coming up because the information is provided by volunteers, and criticism is arising concerning the accuracy and reliability of the provided information (Jia, 2010). Nevertheless Giles (2005) found out that both Wikipedia and Encyclopedia Britannica have similar level of accuracy and the frequency of severe errors is alike. Results from Wikipedia shown that it is possible to accumulate good quality data but the errors are unavoidable (OpenStreetMap, 2010b).

VGI is based on wiki technology as well and is a user generated geographic information content, and for which each volunteer is a sensor and is collecting geographic data, uploading the data to the server, editing the data uploaded or edited by other users and downloading the data for personal use, without considering that the user is a GIS expert or amateur. As for Wikipedia some issues are arising concerning the VGI such as data uncertainty or accuracy, human privacy, etc (Jia, 2010).

Concerning the accuracy there are two main problems, first is that the data collected by the users is being forwarded straight to the database without quality strict inspection over it and secondly the control over the correctness of the data, which is impossible due to diversity of the sources and volunteers' experience in geographic information.

Because most maps have restriction on their use, OSM is giving to people opportunity to use needed map data in a more productive and creative way. For the same purpose OSM has no restriction for the type of data, which can be uploaded to database as long as it is accurate and is not violating anyone's copyright. OSM current main task is to expand the coverage of map data without using existing maps (OpenStreetMap, 2010b). The OSM data is under agreement of "Creative Commons Attribution Share-Alike 2.0 license (CC-by-SA)" open source license.

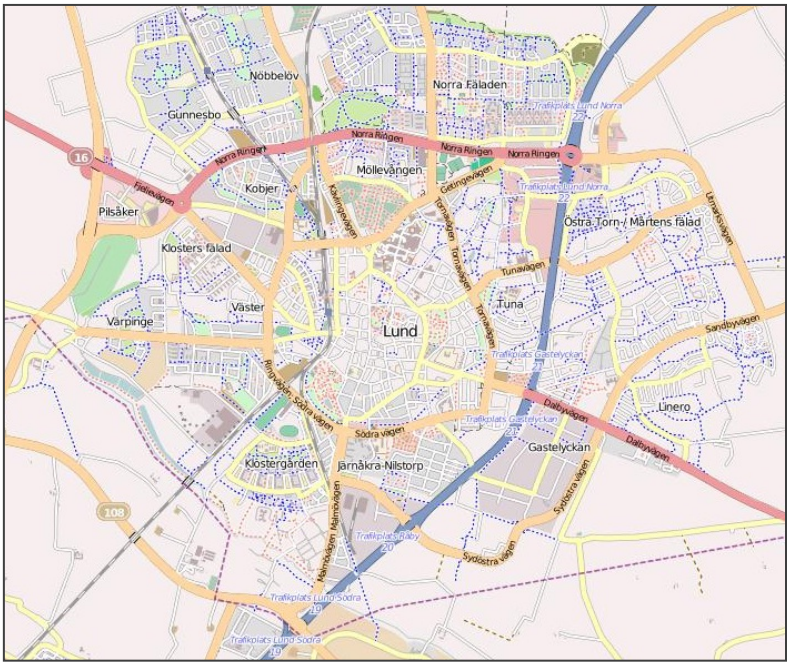


Figure 3.2. OSM map data of Lund city (OpenStreetMap, 2010a).

The OSM map data for Lund city is shown on Figure 3.2. As the completeness of data varies all over the world and the completeness of the OSM data for different areas can be observed from different empirical analysis done by different authors. In the empirical analysis of pedestrian road network for Germany done by Zielstra and Zipf, (2010) the relative completeness of the road network was determined by the comparison of the total length of the road network data from OSM with the length of the road network data from other professional geographic data providers. And from the analysis it can be seen that the total length difference (in percents) of pedestrian related data (smaller ways, alleys, streets, etc.) between OSM and Tele Atlas where OSM data was taken from different time periods (Figure 3.3). The study shows that the length difference was reduced more that 25% in one year period. And because OSM data is updated

almost every day; more complete pedestrian and road network data is expected in the future that would be useful for more and more applications.

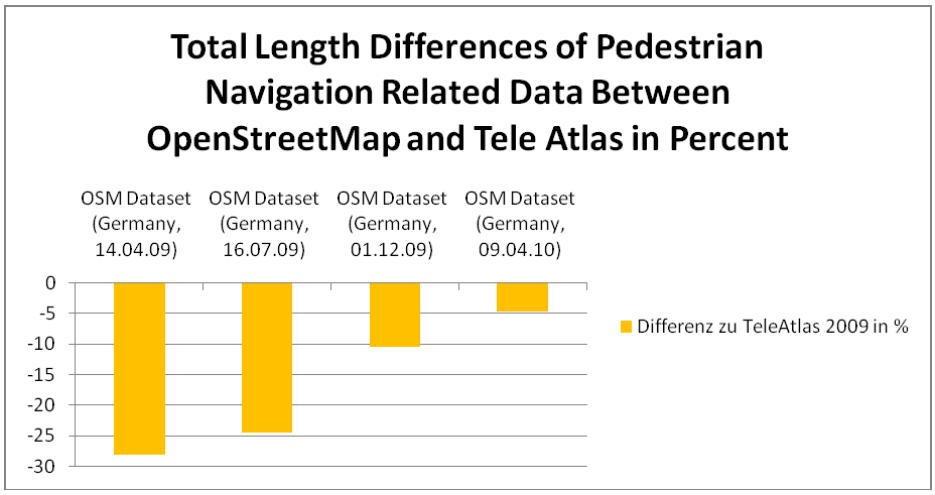


Figure 3.3. Comparison of OSM and TeleAtlas pedestrian navigation data, where OSM is taken from different time periods (Zielstra and Zipf, 2010).

3.3.2. NVDB (*Nationella Vägdatabases*)

Swedish national road database (NVDB) is authorized by Swedish government and includes all Swedish roads network and selected cycle paths (NVDB, 2008). NVDB includes all squares, streets, roads, ferry routes and also feasible to enter cycle paths but no pedestrian paths are included in it. NVDB is providing raw data for multipurpose applications and it does not include any services in it.

The NVDB main contributors are the Swedish transport administration, Lantmäteriet, local municipalities and forest companies. Lund Municipality provides the municipality road data to NVDB. The NVDB data are coordinated by Swedish transport administration and it is required to purchase a license to use the data.

NVDB data is available in four different formats: NVDB's internal data format (*.nvdb), XML based SS637004 Swedish standard format (*.xml), ESRI Shape file vector format (*.shp) and ESRI Personal GeoDatabase which is a Microsoft Access database with supplementary group of tables storing the metadata (*.mdb).

For NVDB data creation two parts of construction principles exists: first part includes road positions related to the landscape and their connections and second part includes also road properties and traffic rules. There are three levels of details exist for viewing roads information (cf. NVDB, 2008):

- Road level: In this level all roads are represented by one reference line irrespective of the number of lanes it includes.
- Carriage level: In this level each separated physical lane is represented by one reference line.
- Traffic level: In this level every lane is represented by its own reference line and also features. Intersections can be represented in this level.

The quality information of NVDB data is modeled according to Swedish standards for geographic information which itself is based on ISO standards and CEN (the European Committee for Standardization). It includes full car road networks for all over Sweden, and also includes cycle paths for selected areas. The NVDB data are created to be used in multipurpose applications like emergency services, taxi and tourists navigation, transportation planning, planning traffic security, roads operation and maintenances, school transportation, transportation for disabled and home-help services (NVDB, 2008). As a national road database the road network lineage in details is described in “NVDB Contents – Feature Types” specification.

CHAPTER 4. Technical Solutions for Web and Mobile Mapping

4.1. Storage of Geographic Data

4.1.1. Overview

Generally there are four fundamental methods of storing geographic data (Figure 4.1). The first type is the case where relational database approach applied to store attributes and non standard format are used to store geometric data. Some products from ESRI and MapInfo are using this approach; for instance the shape file format is composed from two files: *.shp file which stores geometry and *.dbf file which stores attribute. For the second type additional program is used to add functionality to handle geometric data on the top of the relational database in the way of extended SQL. An example for this type is ArcSDE developed from ESRI. In the third type the functionality to handle geometry is added on database stage and as for second one interface is the same extended SQL, for ex.: Oracle Spatial, PostGIS. The last and fourth one is based on object-oriented databases. Non standard query language is the interface to the database; examples adhering this approach are *1spatial* and *Smallworld* (Harrie, 2008a).

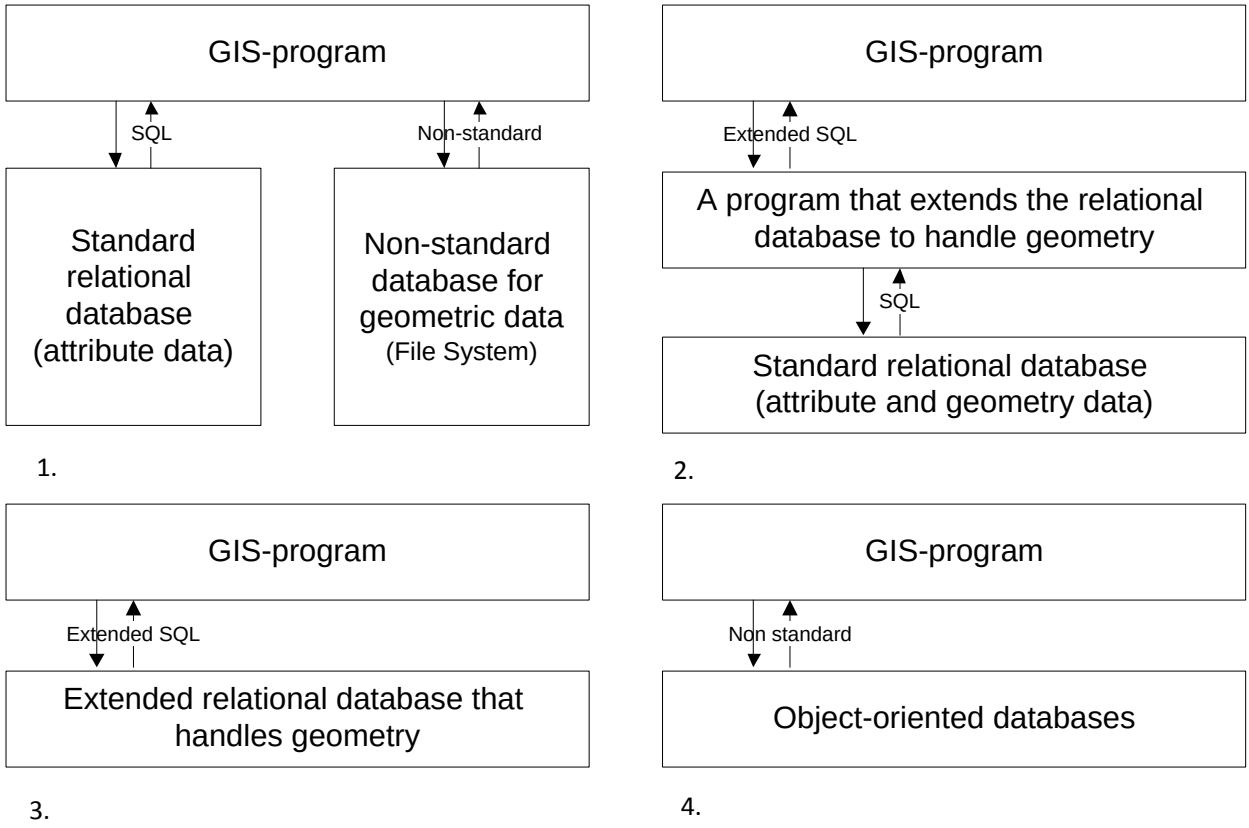


Figure 4.1. Database solution types (Harrie, 2008a, p. 15).

4.1.2. File Systems and Databases

Data can be stored on data storage devices (hard disk drive, CD disk, etc.) as a file in file system or in database. A file system is a technique to store and organize computer files for storage and retrieval. File system consists of root - directories and sub directories and they are structured in a hierarchical tree structure where the files are placed in sub-directories in the tree. The advantage of the file system is a human-readable hierarchical organization and easy manipulability of the data. It is possible to store geographic data in the database, or in file system in either text based formats (XML, GML etc., see section 4.2.1 - 4.2.3) or in binary format.

A database (DB) is a computer-based set of data, structured in a unique way, which can be served to certified users, and is responsible for: organized structure, access, manipulation, retrieval and presentation by use of a piece of software called Database Management System (DBMS) (Figure 4.2) (Worboys and Duckham, 2004).

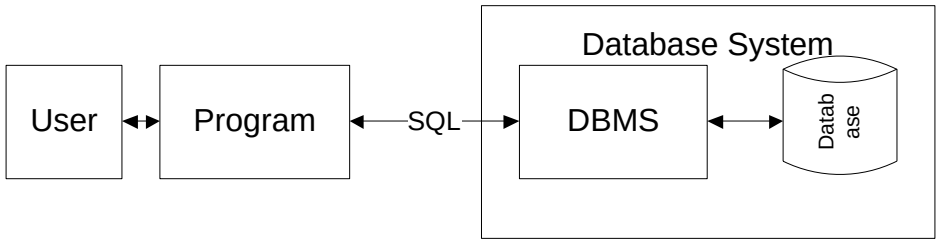


Figure 4.2. User database interaction schema.

DBMS allows specifying a structure of the database. Data sharing is an important element of database ideology. Mainly all DBMS's are providing data-sharing functionality by providing different level of access to different classes of users. To keep the database healthy the Database Management System (DBMS) is following certain requirements which are the advantages of using database systems, and these requirements are (cf. Worboys and Duckham, 2004):

- Security: Unauthorized users should not have access to the data.
- User views: User views should be flexible; each type of user should have different levels of access to the data.
- Reliability: It should be prevented any loss of information in unexpected situations like power failure.
- Integrity: The users should have guarantee that the data is accurate.
- Independence: Some users don't care how the database works, and they should not have a low level access.
- Metadata support and Human-database interaction: Users should have some mechanism to see the database content and retrieve information according to their obligations.
- Performance: The data retrieval should be as fast as possible.

- Concurrency: The mechanism should control the transactions which are proceeding in the same time using the same data, for preventing unexpected data loss.
- Distributed systems: The store of data should not necessarily be physically centralized.

4.1.3. Relational Databases

The Relational Database (RDB) has a simple structure based on group of tables (tabular relations). Because of its minimalism it is so powerful and the majority of databases are today relational.

In the tables the columns stores the attributes, and in the tables the information is kept in rows (tuples) where each row contains values (attributes). The order of the rows (tuples) are not important and no rows are associated (Table 4.1).

Table 4.1. Simple table (relation) example.

RoadID	Road_Name	Road_Type	Road_Length	Max_Speed
1	Drottensgatan	One-way	130	30
2	Stara Tvärgatan	Two-way	260	50
3	Mårtenstorget	One-way	500	30
…	…	…	…	…

The attributes in table (relation) are connected to a certain domain, similar to data type like double, string or date (Worboys and Duckham, 2004). In traditional databases the domains are limited to text and numbers which is the reason why relational database is incomplete for spatial database requirements. Here are definitions of relational database main terms (cf. Shekhar and Chawla, 2003):

- The relation is called to be in first normal form when the data values are atomic (values cannot decompose to set of values or arrays) (Worboys and Duckham, 2004).
- In relational database each table (relation) should contain a primary key.
- A key is a subgroup of the attributes whose values can matchlessly identify the row (tuple) in the table (relation).
- The one key from several keys in the table (relation) which can identify the rows is called primary key.
- A foreign key in the table (relation) is the one which is primary key in other table (relation) (Shekhar and Chawla, 2003).

4.1.3.1. SQL

In an RDB, the user communicates with the DBMS through the Structured Query Language (SQL). SQL makes connection between user and a relational database to state a database scheme and allows the data manipulation, i.e. to input, modify and retrieve data.

Data Definition (or Description) Language (DDL) and Data Manipulation Language (DML) are two subsets of SQL (Figure 4.3). DDL part of the SQL provides functionality to create, modify and remove data scheme in a relational database. By use of DDL user can also create a domain. The DML part of SQL provides a set of data manipulation operations like *intersection*, *union* and *difference* along with operations *join, project, restrict* and *divide*. The most complex part of SQL is data retrieval; the simple structure is as follows (Worboys and Duckham, 2004):

SELECT *list of attributes for output table*
FROM *relation reference*
"WHERE *condition*" [optional]

A simple example of query from the Table 4.1 can be to retrieve the length of the road *Drottensgatan* and the structure of the query will be as follows:

```
SELECT roadnetwork.RoadName, roadnetwork.RoadLength
FROM roadnetwork
WHERE roadnetwork.RoadName = 'Drottensgatan'
```

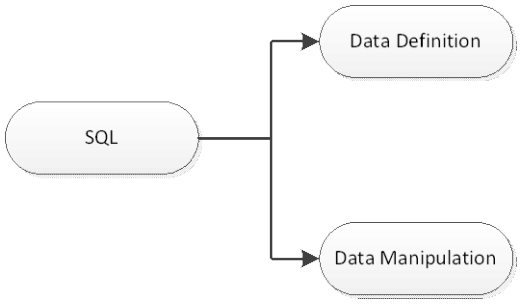


Figure 4.3. The subsets of SQL

4.1.3.2. Indexes

Data can be stored in datasets either ordered or unordered. It is easy to create and modify unordered dataset, however the search is performed much slower compared to the ordered datasets. The computational complexity of the search performed on the unordered dataset is O(n) as the linear search is performed; compared with the ordered dataset where the binary search is takes O($\log_2 n$) time. To improve the search time indexes are introduced.

"An index is an auxiliary structure that is specifically designed to speed the retrieval of records" (Worboys and Duckham, 2004, p. 225). In other words an index is an ordered copy of all records in one field of the dataset, and binary search will be performed on that field. It is preferable to create index only for the fields which are queried frequently, because indexes pursue additional computations and space on the disk. As geographical data does not have natural alphanumerical order additional space-driven and data-driven access methods are introduced (Harrie, 2008a).

4.1.4. Spatial Databases

Problems occur when relational databases are applying on spatial data. There are three fundamental problems when we want to apply relational databases on spatial data: structure of spatial data, performance and indexes. For handling spatial data more complicated structures than relational databases are essential with additional operations and index methods to support these complex structures (Worboys and Duckham, 2004).

The structure of spatial data is physically incompatible with relational database structure. An example is vector areal data where the edges composed of line segments structured in order, and each of these segments consists of points, thus it goes against first normal form of relational databases. Also for building spatial object there is a need to join numerous relations and this essentially affects the performance. Usually it is needed to rapidly retrieve spatial data for example for showing on the screen. As a result relational databases do not permit to reach high performance while handling spatial data.

For facilitating relational databases to handle spatial data extensible relational databases (ERDB) were introduced. "An extensible RDBMS is designed to provide specialist users with the facilities to add extra functionality, specific to their domain" (Worboys and Duckham, 2004, p. 54). As it was intended ERDBs extends RDMS (ERDMS) and allows definition of abstract data types (ADT), operations on the data, specific indexes and access techniques and active database functions; most of these targets ERDB shares with object-oriented databases (OODB) (Worboys and Duckham, 2004). For spatial data general ADTs are *point, line* and *polygon*. It is very important to understand that physically for instance point is stored as x and y (and optional z) coordinates, however ADTs should be considered as logical units. Additionally ERDMS allows operations on ADTs, for example an operation can be implemented to calculate the area of the polygon, and this operation allows polygon as an argument, and these simplifies SQL queries (Harrie, 2008a).

Spatial databases require specific index structure. Because computer storage is one dimensional but spatial data is expressed in multidimensional Euclidian space and has no order, space filling curves (example: Morton key) as space-driven access methods and data driven methods (example: kD-tree) are introduced. These methods are implemented to reach acceptable

performance, however not all RDMSs support these indexes (Shekhar and Chawla, 2003). A thorough description of these methods is given in Worboys and Duckham, 2004.

4.1.5. PostGIS

Refractions designed and developed PostGIS as an open source project. “PostGIS adds support for geographic objects to the PostgreSQL object-relational database” (PostGIS, 2010b). PostGIS supports the Open Geospatial Consortium (OGC) “Simple Features Specification for SQL” standard for “extended SQL”. A few open source tools are available for operating with PostGIS (e.g. uDig) which allows full reading and writing functionality in on-screen environment (PostGIS, 2010a). It is also possible to work with PostGIS with any programming language that PostgreSQL supports as C++, Java, C#, etc.

PostGIS Features are: high performance, data integrity, spatial query and spatial analysis (PostGIS, 2010b). PostGIS is drastically increasing performance by use of storing geometries in minimum possible representation. It comes out from PostGIS users that PostGIS shows top performance with working on a huge spatial data. Minimum representation enhances performance directly by making possible to maintain data in fast memory cache (PostGIS, 2010b). Spatial data database maintenance makes it possible to access data with any tool which supports SQL. During processing spatial data PostGIS prevents resource discrepancy and assure data integrity by implementing row-level locking. As mentioned before PostGIS supports “Simple Features Specification for SQL” standard thus supports complete set of geometry query operations like intersection, distance, etc. and complete relationship matrices, and additional to this R-tree indexing affects on the query quickness. Advance spatial analysis can be implemented by use of spatial joins, affine transformations, buffers, etc. (PostGIS, 2010b).

4.2. Distribution Formats

4.2.1. XML

XML (Extensible Markup Language) is a text based file format, which is created for storing and transporting data. The data stored in XML document is structured in elements like <road>, <name>. XML allows creating as many elements as it is needed for certain data storage.

In Example 4.1 a simple XML document is shown. Every element in XML document has start tag as <road> and end tag as </road> and the text in-between this two tags is called element content. Overlapping tags are unacceptable in XML, example is <road><name>Mårtenstorget</road></name>.

In the Example 4.1 there is only one *road* element. There are four subelements to *road* element and they called *child* elements of *road* element (Figure 4.4). The *road* element is called *parent*

element for that four elements included in it. The *name*, *length* and two *restriction* elements are called *siblings*. Each *child* element in XML document has only one *parent* element. There is only one element without *parent* element in XML document, and it is called *root* element. In our example the root element is the *road*. It is a requirement to have one and only one *root* element in XML document (Harold and Means, 2001).

It is possible to create new elements while creating XML document. It is also acceptable to have empty tags, but it can be formatted as previously as <road></road> or just open and close the tag in the same time road <road/>.

XML is case sensitive, that means that <ROAD> and <road> will be read by computer as a different tags. It is possible to have both of them but it is not a good style of having them in the same document.

```
<road>
        <name>Mårtenstorget</name>
        <length>500</length>
        <restriction>one-way<restriction>
        <restriction>30<restriction>
</road>
```

Example 4.1. Simple XML document.

XML documents are structured like a tree. Because of overlapping restriction in XML document and that all elements have only one parent element (exception is the root element); XML document can be represented as a tree (Figure 4.4) (Harold and Means, 2001).

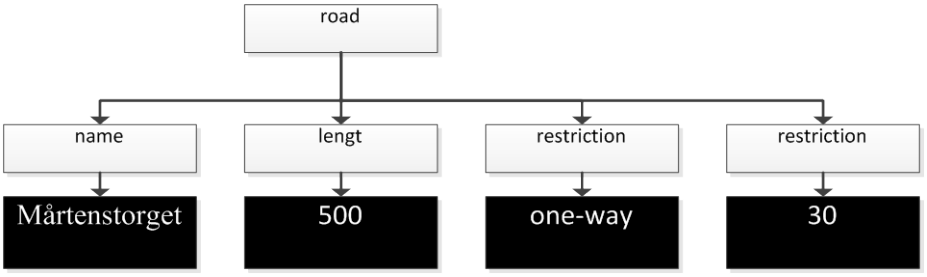


Figure 4.4. Tree diagram.

XML supports attributes. An attribute is supplementary information about an element which should be stated in start tag. Attributes are defined by name and value, as <length unit = “meter”>500</length> in Example 4.2. In this example element *length* has attribute *unit* which has *value “meter”*. The values should be quoted in single or double quotes; sometimes it is better to put in single quotes because value itself can have double quoted value (Harold and Means, 2001).

```
<road>
        <name>Mårtenstorget</name>
        <length unit = “meter”>500</length>
        <restriction type = ”roadtype”>one-way<restriction>
        <restriction type= “maxspeed” unit = “km/h”>30<restriction>
</road>
```

Example 4.2. Simple XML document with attributes.

To summarize, the fundamental requirements for XML document to be well formatted are (Harold and Means, 2001):

- There is one and only one root element.
- Every start tag has an end tag.
- All open tags are closed.
- No nested elements are allowed.
- Attributes are quoted.

It is subject of discussion when to use attributes and when to use elements for storing information. It is obvious that elements are the right solution when it is needed to store more than one additional information with the same name like *restriction* in our example. Attribute stores a simple text string as a value. Elements are more convenient, adaptable and extensible; however, attributes are preferable for certain applications (Harold and Means, 2001).

4.2.1.1. XML schema and DTD (Document Type Definitions)

XML is a meta-language for defining mark-up languages for certain applications. Some examples of XML based languages are: GML (Geography Markup Language), KML (Keyhole Markup Language) and SVG (Scalable Vector Graphics). Each XML application parses the XML base document and follows the rules for certain XML base language, ex. GML, KML and etc. (Harold and Means, 2001).

XML schema is a depiction of XML document type. XML schema includes restrictions for the structure and content of the XML document. The restrictions are expressed in the term of grammatical rules restricting the order, privilege and occurrence of elements and attributes. Many languages are developed to state XML schemas, examples are: Document Type Definition (DTD), XML Schema, etc.

XML validation is the procedure when the parsers is checking XML document well-formedness and comparing the structure of the document to the declaration defined in certain DTD or XML

schema and lists mismatches. If the parser doesn't find any mismatches it means that the document is valid. The simple structure of a DTD file for Example 4.1 is shown in Example 4.3.

```
<!ELEMENT road (name, length, restriction*)>
<!ELEMENT length (#PCDATA)>
<!ELEMENT restriction (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST length
unit    CDATA        #REQUIRED>
<!ATTLIST restriction
type    CDATA        #REQUIRED
unit    CDATA        #IMPLIED>
```

Example 4.3. DTD structure for Example 4.2.

Valid XML base language document includes a location of a DTD to be compared. The example of the declaration is: *<!DOCTYPE road SYSTEM "C:/road.dtd">*. It includes root element and the path to the DTD file. The declaration appears in a XML file after the statement of XML version but before the root element (Harold and Means, 2001). The location also can be a URL (Uniform Resource Locator).

4.2.2. GML

GML (Geography Markup Language) is an XML based language for storing and distributing geographic data. GML stores geographic data in a text structure and thus it can be easily read, modified and transformed like any other XML based file format. Some applications use GML to store data but it is not reasonable in other applications because text files use substantial space (Turton and Dutton, 2009).

GML was originally developed by OGC (Open Geospatial Consortium) and is accepted by greater part of GIS vendors in the world (Lake, 2010). For creating maps from GML data other XML based display formats are used like SVG, X3D, etc.

GML is developed to support compatibility between different data models and feature definitions and a tool for constructing and distributing application schemata (Peng and Tsou, 2003). In GML it is possible to create new markup tags and elements for certain application where is it needed to define spatial features and geometries. GML describes the geographic world by use of units named *Features*. All the elements and attributes affordable in GML are defined in certain DTD linked to a GML file.

GML includes all the spatial and non-spatial information about features and geometries, and also contains spatial reference system of the data. Furthermore, the problem associated with transporting GML data is the size of the files, because GML is text based file format and includes detailed definition of features, the size of the file is becoming very large (Peng and Tsou, 2003).

GML2 supports *Point, LineString, Polygon, Multipoint, Multipolygon* and *Multilinestring* geometries (Turton and Dutton, 2009):

- Point geometry characterizes single point with 2 coordinates in geographic space and elevation as an optional coordinate.
- LineString geometry characterizes linear geographic element: number of Points connected by lines, and coordinates can have elevation as for others.
- Polygon geometry characterizes circumscribed area in geographic space. Like in Point geometry Polygon coordinates also includes optional elevation coordinate. It can have holes, holes can have islands but holes can't reach outside edge of the Polygon.

In addition to GML2 geometries GML3 supports *Curve, Surface* and *Coverage* (Grid) (Turton and Dutton, 2009):

- Curve geometry is an extension to linear element and characterizes even curves like Arc, CubicSpline and Bezier.
- Surface geometry is an extension to Polygon and characterizes geometries like Patches (polygon whose edges are curves), Rectangles and Triangles.
- Coverage is a technique to characterize raster data in GML.

Features are the groups of properties (name, types and value descriptions) and geometries as Point, LineString and Polygon in GML1 and GML2, and additional Curve, Surface and Coverage in GML3 (Lake, 2010).

4.2.3. KML

KML (Keyhole Markup Language) is an XML based language which is developed to store and transport illustrations of geographic data for earth browser (KML, 2010). KML is developed by Google and version 2.2 is accepted as an OGC (Open Geospatial Consortium) standard. KMZ file format is an archive format for KML documents and images linked to it (KML, 2010). With the use of KML it is possible to create Placemarks for identifying POI's (Point Of Interest) on the earth: factories, schools, bus stops, etc.

Google Earth is the most used tool for creating KML document; it allows creating routs, image overlays, photo overlays, etc. The XML schema of KML format is shown on Figure 4.5 (KML, 2010).

KML is developed not only for representing geographic data but also for visualization online and mobile maps visualization (Google maps and mobile Google maps) and also earth browsers (Google Earth). KML includes not only geographic visualization but also regulation of user's navigation: including direction information and so on (KML, 2010).

It is supplementary to GML, and KML v. 2.2 contains geometry elements from GML 2.1.2: point, linear ring, line string, and polygon. In the future Google and OGC accepted to additionally synchronize KML and GML (KML, 2010).

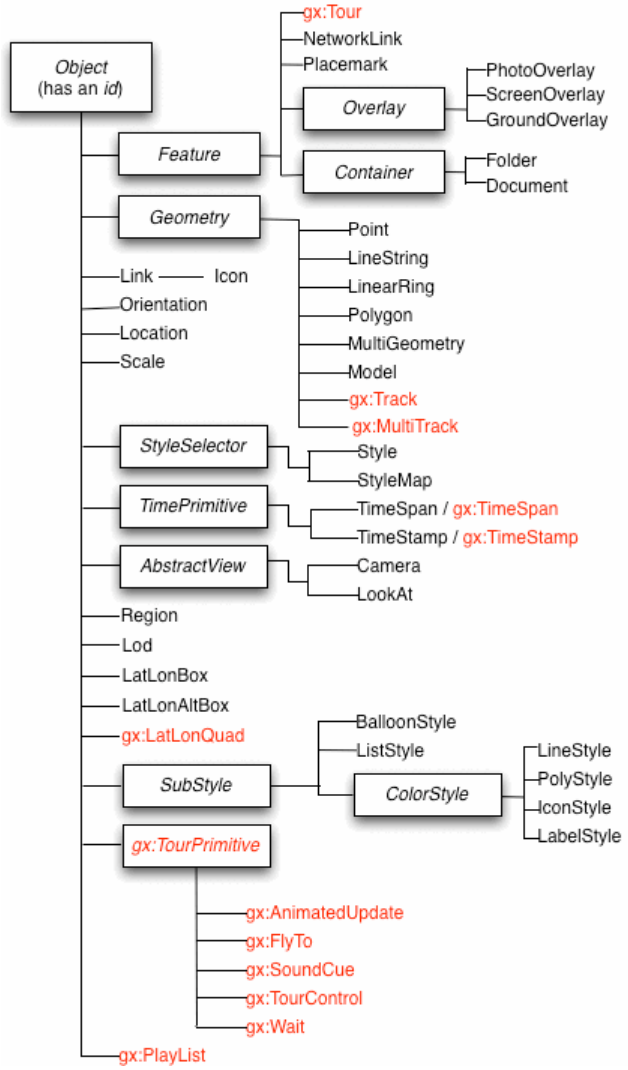


Figure 4.5. XML schema for KML file format (KML, 2010).

4.2.4. GeoJSON

"GeoJSON is a geospatial data interchange format based on JavaScript Object Notation (JSON)" (Butler et al., 2008). GeoJSON is text based format and can be read and edited by any program

which allows text editing. It has a more compressed structure than XML based formats (CityGML, KML, X3D, etc.) (GeoJSON Wikipedia, 2010). There are three so called objects that GeoJSON includes: *geometry*, *feature* and *feature collection* (Butler et al., 2008).

GeoJSON file itself is a single object (*geometry, feature* or *feature collection*). An object in a GeoJSON is a combination of a name and members, where name is a string and member's value is either *object, number, string, array* or values like "*true*", "*false*" or "*null*". There is no limit for number of members in an object. The member with a name "type" is required, and represents the type of a GeoJSON object. There is an optional member called "*crs*", which represents the coordinate reference system and must have value of one of the Coordinate Reference System Objects. Another optional member is "*bbox*" member with a value of bounding box array (Butler et al., 2008).

Geometry is an object which has a value of geometry type: *Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon* and *GeometryCollection* (Butler et al., 2008). All the geometry types, excluding GeometryCollection, have member called *coordinates*. The value of a *coordinates* member is an array. Position itself is an array of numbers: *x, y, z* and others if needed. In this array z is altitude; x, y are easting, northing or longitude, latitude respectively for projected coordinate reference system and geographic coordinate reference system. The member *coordinates* consists of positions: Point geometry – one position, MultiPoint – array of positions, LineString – two and more positions in an array (Example 4.4), MultiLineString – array of LineStrings, Polygon – array of LineRings (LineRing is a LineString where the first and last positions are identical and should contain more than 4 positions) and MultiPolygon – array of Polygons. Geometry Collection has required member "geometries" and it is an array of geometry objects (Butler et al., 2008).

```
{  "type": "LineString",
   "coordinates": [ [ 0.0, 0.0], [10.0, 10.0] ]
   }
```

Example 4.4. Coordinates and positions for LineString geometry type.

Feature itself is a combination of a geometry object and properties. Required member "geometry" should contain or any of previously mentioned geometry object types or a JSON null value and also mandatory "properties" member should contain any JSON object or JSON null. If the feature has identifier it should contain "id" member. Feature collection is a group of features and has mandatory member called "features", which is an array of feature objects.

The coordinate reference system as mentioned above is specified with member called "crs". When the crs member is not stated, it is referred to parent or grandparent crs member; otherwise it takes a default crs which is WGS84 datum with decimal degrees as a unit for longitude and

latitude. As a value for crs member it acquires JSON object or JSON null. It should be stated on the top of the GeoJSON document and should not be repeated or stated again in children or grandchildren objects. The crs member includes two required members when it is not null: “type” and “properties”. The type member acquires value string to specify crs member’s type and the member properties acquires object as a value. It takes “name” as a value when the coordinate reference system is stated in the object, that time the properties member should contain member called “name”. As a value name member acquires string, which defines the coordinate reference system (Example 4.5). It is also possible to give a link to a coordinate reference system for that purpose type member should have value “link” (Butler et al., 2008).

```
"crs": {
"type": "name",
"properties": {
"name": "urn:ogc:def:crs:OGC:1.3:CRS84"
}
}
```

Example 4.5. Code for definition of coordinate reference system in GeoJSON (Butler et al., 2008).

The bbox member of GeoJSON object is used when it is necessary to specify coordinate ranges for GeoJSON objects. The bbox member is 2*n dimensional array, where n is a number of geometries dimension (Butler et al., 2008).

4.3. Client-Server architecture

GIS applications, as well as several other applications, consist of three fundamental components: data, logic and interface (Figure 4.6). Where the *data* component represents the database, the *logic* represents the processing component and finally *interface* component refers to User Interface (UI). Using UI of the GIS software users perform actions for example queries or overlays which logic component is processing whereas logic component generally is relying on data in the database to generate the results (Peng and Tsou, 2003).

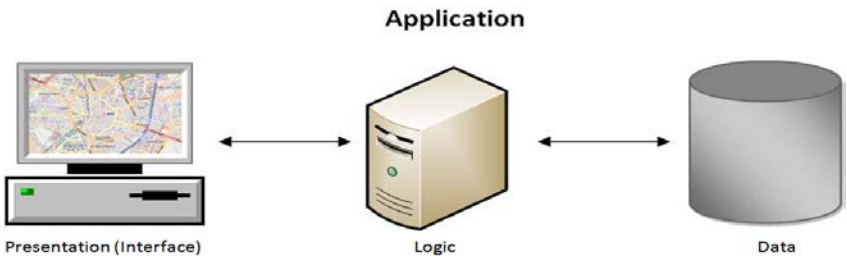


Figure 4.6. Components of a general application

The application is called stand alone when all three components are residing in one machine. When these components reside in several machines and one or several machines are responsible for each component at that time the application is using client-server model and is called client-server application (Peng and Tsou, 2003).

The client-server architecture has distributed application structure which separates tasks of the application between different information systems. The information system that provides the services to other information systems within a network is called server. While the information system that uses the services is called client (Worboys and Duckham, 2004).

Generally client-server architecture uses request-response protocol, for example HTTP (Hypertext Transfer Protocol) works as a request-response protocol using client-server architecture where the client (web browser) is sending web-page request to the server and getting back the response as a web-page or files and delivering the result to the user. The protocol is a communication standard format (Worboys and Duckham, 2004).

The main distinctiveness of the client-server architecture is the partitioning of the process responsibilities between client and the server. Thus the terms thick and thin client and server are introduced. The server is qualified as thick when server is performing substantial processes to accomplish the task and here we have thin client. And the opposite case is the thick client and thin server, where the client is performing the substantial processes to accomplish the task (Worboys and Duckham, 2004).

Following examples are special cases which are used in different GIS application solutions, starting from very thin client to very thick client (Peng and Tsou, 2003):

In the case of *very thin client* two components: logic and data, and a part of the client are placed in server, only the mirror of the result is passed to UI. An example is the case when GIS software is running in the server and only the mirror window is accessible to the user.

Thin client is the case where the presentation is located in client side but logic and data are located in the server and example is static maps displayed using web browser.

Medium client is dividing the logic between the server and the client, and some processes are operating in the client and some complex process are requested from the server and example is web maps presented in the browser and on the action of zoom-in: the higher level map is requested and the respond of the server are tiles.

In the case of *thick client* logic and presentation components are located in the client and the data is located in the remote database. Example can be GIS application which is performing all the functionalities in the client side and only querying the data using SQL from the data server.

And at last *very thick client* is the model where presentation, logic and a part of data management system is implemented in the client side and only data and a part of functionalities for data management is located in the server. GIS applications which are capable of rendering data from distributed databases are good example of applications using very thick client model.

The advantage of the client server application on the contrary to mainframe application (stand alone) is that the client can use services from several servers. In the client server architecture all users requests should be submitted to the server and due to the network traffic responds will be delayed. However this issue can be solved by simply adding another data server, also additional backup server will prevent the user application failure in the case of server failure (Peng and Tsou, 2003).

The advantage of the server side strategy is that the data processing is being performed in the server where respond contains already processed data instead of sending all the data needed for processing, an example can be routing request, and the respond will be the part of the network which is the route however all the network data would be used for routing calculation. Also an important aspect of the server side strategy is the data security that all the data is being hold by service provider (Worboys and Duckham, 2004).

In respect of client side applications they are preferable for experts as they demand more functional and flexible applications, however in the case where huge amount of non experienced users are exploiting the application, server side strategy is more suitable as data is stored in the server and is secure and the processing of the data is done in the server and only simple functionalities are available to the user (Worboys and Duckham, 2004).

4.3.1. Navigation Applications on PND's, PDA's and Smartphones

Nowadays most common solution for PNDs (Personal Navigation Device) is stand alone architecture where all three components: presentation, logic and data, are residing in the PNDs. However most PNDs vendors tend to use very thick client model of the client server architecture, where the main part of the data (road network, POIs, etc), logic (routing calculation, user interactivity, etc.) and presentation components are inbuilt in the client and just additional information which are necessary for advance navigation are requested from distributed databases like real time traffic situation, map updates, etc.

The active competiting sector is navigation enabled smart phones and mobile devices with inbuilt GPSs. According to TomTom (2011) smartphones and PDAs will exploit more and more navigation applications and services and will be used for daily navigation however they will not replace the services that PNDs are providing and will provide.

Because of rapid development of smartphones and PDAs; depending on their characteristics there are available variety of free and commercials navigation applications. And they are using different application solutions from thin-, medium-, thick-, very thick client applications to stand alone application.

4.4. OGC Standards

4.4.1. Web Map Service (WMS)

Map services are implemented to delimit functionalities among client and server. OGC has defined several mapping services for different needs.

"A Web Map Service (WMS) produces maps of spatially referenced data dynamically from geographic information" (WMS, 2011). A map does not constitute the data itself moreover a *map* is a depiction of geographic information in digital image file format - defined by ISO. The maps produced by WMS are usually depicted in digital image format such as Graphics Interchange Format (GIF), Portable Network Graphics (PNG), Joint Photographic Experts Group (JPEG) or as a vector based graphical formats: Scalable Vector Graphics (SVG) and Web Computer Graphics Metafile (WebCGM) format (WMS, 2011).

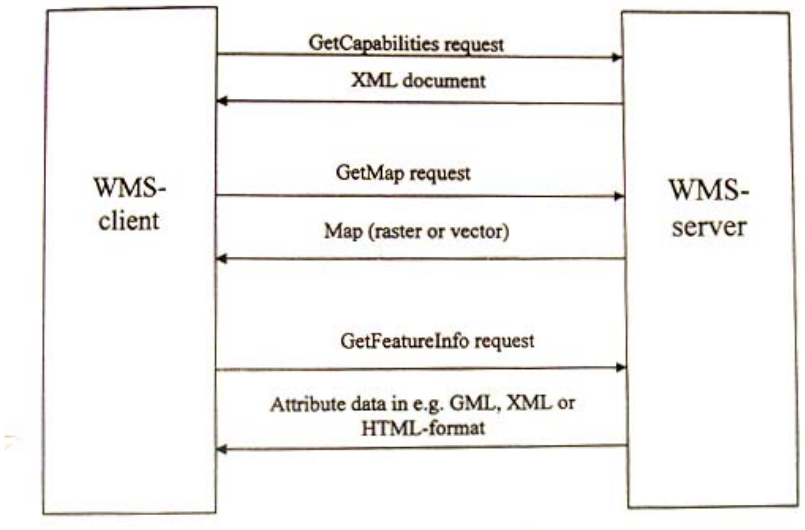


Figure 4.7. Communication between WMS client and WMS server (Harrie, 2008a)

The WMS performs three operations (Figure 4.7) (WMS, 2011):

- *GetCapabilities* (requesting service-level metadata)
- *GetMap* (requesting a map with defined metadata)
- *GetFeatureInfo* (optional) (requesting information of some features appeared on a map)

The operations of the WMS are possible to execute by submitting Uniform Resource Locator (URL) request where the requested operation will be performed depending on the content of the URL. The content of the query also defines the metadata such as bbox of the map, coordinate reference system, etc. Image formats: GIF and PNG, support transparent background and this allow overlaying of several maps from the same or different servers with identical geographical parameters and covered area by map (WMS, 2011). The WMS

4.4.2. Tile Map Service (TMS)

A Tile Map Service (TMS) provides the specification to store and retrieve cartographic maps of geo-referenced data. It standardizes the client request of map tiles. The Tiled Web Service allows rendering cartographic tiles at defined scales. The OSM project is utilizing this standard. In contrast to TMS, the WMS standard provides high level of flexibility (TMS, 2011).

4.4.3. Web Processing Service (WPS)

"WPS defines a standardized interface that facilitates the publishing of geospatial processes, and the discovery of and binding to those processes by clients" (WPS, 2011). WPS describe rules to formulate standards for processing/analyzing spatial data and thus it is a "meta-standard" (Harrie, 2008b). WPS allows provision of any GIS functionality across a network, whereas data can be stored at the server or can be provided by client. The WPS designed for processing both raster and vector data. An example of WPS is Web Routing Services, where the client just provides the start and destination position and gets the route as a result; does not having knowledge about the processes performed at the server side (Figure 4.8) (WPS, 2011).

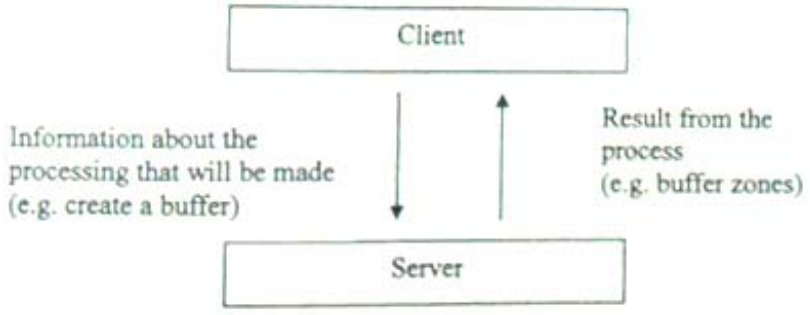


Figure 4.8. Communication between WPS client and WPS server (Harrie, 2008a)

4.5. Client System

4.5.1. Introduction to Android OS

Android is an open source mobile operation system developed by Google. The founder of Android; The Open Handset Alliance was created by collaboration of several organizations “to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience” (Open Handset Alliance, 2011). By authority of Google the group members are mobile operators, handset manufacturer companies, semiconductor companies, software companies and commercialization companies.

Consumers benefit from the open platform as the mobile devices become less expensive and more innovative, and it makes easier the user interface and makes available plenty of applications based on Android platform (Open Handset Alliance, 2011). Google Android Market was established in October 2008 and is an online software store for Android devices (Android Market Wikipedia, 2011). Handset manufactures benefit from the platform because of low software cost and fast time for handset production. Developers their self will benefit as the complete API access to handset is available, increase productivity is guaranteed because of user-friendly developer tools. Additionally the mobile application commercialization is cheaper and easier (Open Handset Alliance, 2011).

The Android SDK (Software Development Kit) is a software development environment that makes it possible to create applications that can be installed and run on the Android platform. Android SDK includes development tools, sample code, desktop emulator for testing and build in libraries that allow Android application development. Android SDK is based on Java programming language while the applications run on virtual machine called Dalvik. The Dalvik Virtual Machine itself runs on Linux Kernel (Android Developers, 2011a).

The Eclipse open source development platform is used for Android application development, while it is necessary to extend it with Android Development Tools (ADT[1]) plug-in for Eclipse. The Android package is one application as an archive file that encloses the compiled code with resource files. The Android package is stored with *.apk suffix (Android Developers, 2011a).

4.5.2. The architecture of the Android platform

The Android platform is divided into four layers with 5 subgroups (Figure 4.9) (Android Developers, 2011a).

Application layer

The Android operation system includes build in applications such as email client, SMS program, calendar, maps, browser, contacts, etc. The applications are developed using Java programming language. All the developed or downloaded applications will run on this layer (Android Developers, 2011a).

Application framework

The application framework provides developers ability to develop reach and innovative applications. It implements the standard structure of the Android application. The application framework is designed to allow the reuse of the components, and any application can reuse the components of other existing applications, and this simplifies the application development (Android Developers, 2011a).

All applications consist of services and systems such as (Android Developers, 2011a):

- Views- which are used to build an application; examples are lists, grids, text boxes, buttons and even Google maps.
- Content providers allows the access of the data from different applications (ex. Contacts) or share own data.
- Resource Manager that allows the access to non-code resources such as graphics, layout files, etc.
- Notification Manager that enables the status bar alert messages.
- The Activity Manager operates the lifecycle of the application.

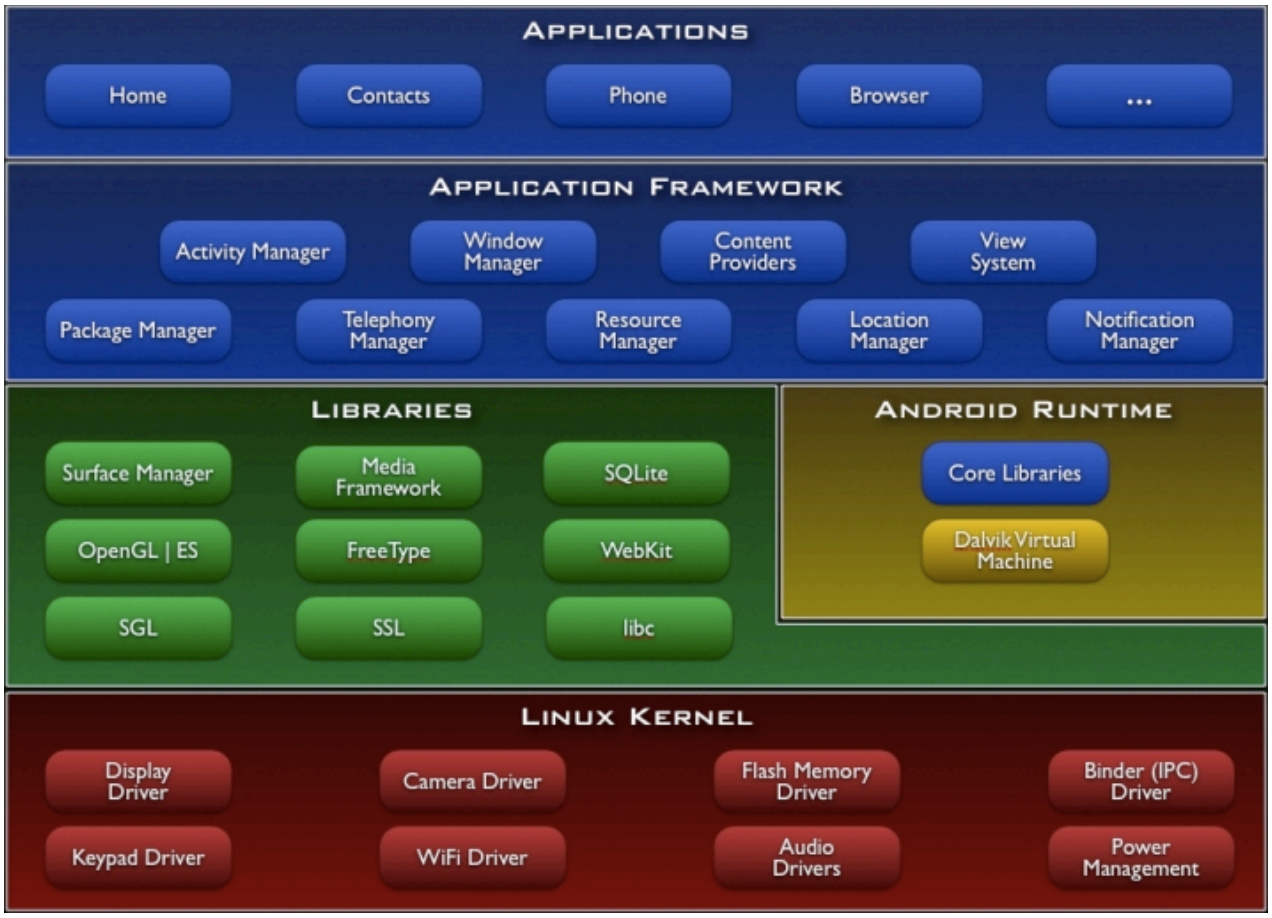


Figure 4.9. Main components of the Android (Android, 2011a)

Libraries

Android contains libraries developed in C/C++. The main libraries are: System C library (to embed Linux-based devices), Media libraries (to support audio, video and image formats), 3D libraries, LibWebCore (web browser engine), FreeType (to support bitmap and vector font rendering), SQLite (SQL relational database engine), etc. The libraries are developed to be used in Java environment (Android Developers, 2011a).

Android Runtime

The Android Runtime has two components; the first component is a core libraries that makes available the functionalities enclosed in the core libraries of the Java programming language and the second component is the Dalvik virtual machine (Android Developers, 2011a).

Linux Kernel

The Android platform is based on Linux Kernel. Linux Kernel provides the system services to Android, namely: security, memory management, process management, network stack, and driver model. The kernel functions as a link between the software stack and the hardware (Android Developers, 2011a).

4.5.3. Android Application Structure

Each application in Android system is running on separate virtual machine that isolates the applications one from another. The Android system is based on the least privilege principle that is each application has permission to defined components that is required for certain job, and all remaining components are isolated from the application. This enhances the security of the Android system. However as it was mentioned before, each application can have permission to system services and another application's data. All the permissions of the application such as permissions to device data (contacts, accelerometer, GPS, etc.) should be defined at install time (Android Developers, 2011b).

Four different application components exist. The components are used to build the application and the use of different components is depending on the application, each one has its own lifecycle and some components are related to another. The four components are (cf. Android Developers, 2011b):

- **Activity**
  The Activity component is a user interface that can cover whole screen or slide on top of another activity. The user can interact with activity such as to take a picture, view a map, etc. Typically the applications consist of several activities and one is defined as "main" activity. One activity can call another activity to perform an operation. All the activities defined in the application are subclass of the Activity class.

- **Service**
  The Service component is running in the background. The services are used when it is necessary to perform long term operations such as playing music in the background and also it is used to obtain the data over the network not interrupting the user-activity interaction. All the services defined in the applications are subclass of the Service class.

- **Content provider**
  The content provider component allows storing and retrieving the data, and allows the data sharing across the applications. The content providers for different data types are supplied by Android. It is possible also to share own data either by creating own content

provider or by adding it to existing provider. All the content providers defined in the applications are subclass of the ContentProvider class.

- **Broadcast receiver**
  The broadcast receiver component receives the broadcast announcement, where the announcements can originate from the system or from other applications. An example of the broadcast announcement can be the system broadcast that the GPS receiver is on, or the data from another application received, etc. The broadcast receiver component does not have a user interface however the status bar notification may pop-up to deliver the outcome. All the broadcast receivers defined in the applications are subclass of the BroadcastReciever class.

The applications or the components developed for Android system is possible to use by another application, thus for example the camera photo capturing component can be used in other application and there is no need to develop it again. The *intents* are used in Android system to send a message to the system to activate the component from other application and the system is activating the component for use. The intents are used to activate the activities, services and broadcast receivers (Android Developers, 2011b).

All the applications should contain in the root directory AndroidManifest xml file. The components that will be used in the application should be declared in the manifest file. In the manifest file should be declared also the permissions such as Internet access, camera, GPS and more hardware and software features such as the minimum API level (Android Developers, 2011b).

Beside the code Android application requires resources such as visual representation (style and structure) of the application, images, audio files, etc. The visual representation of the application is stored in XML files (Android Developers, 2011b).

4.6. Map solutions for Android platform

The Google Maps External Library allows the user to integrate Google maps in Android applications. It provides the functionalities to display the Google map tiles as well as the data derived from other Google map services. The functionalities such as panning, zooming, and the use of other services allows for the interactive use of the maps (Android Developers, 2011c).

The MapView class is used to display the map tiles; it is responsible for all the elements controlling the user interface interaction. In addition, it allows the overlays on top of the base map (Android Developers, 2011c).

As the Google Maps External library is not included in the standard Android library, some devices do not support these functionalities. This library is a Google API add-on which is available for download and supplements the Android SDK. In order to use the Maps API it is necessary to register and get the Maps API Key (Android Developers, 2011c).

There are several open source libraries that allow easy integration of different maps solutions in Android mobile applications. The Nutiteq is a company that provides mobile map solutions for different mobile platforms which are used by mobile service providers in Europe, US and Middle East.

The Nutiteq Maps Lib SDK for Android includes the Google Maps package component; additionally it allows the integration of maps from different sources such as: OSM, CloudMade, BLOM aerial, Bing/Microsoft, Navteq/MapTP, DigitalGlobe aerial and Yahoo maps. Beside the previously mentioned map services, it is also possible to integrate maps from user-created map services by using WMS and other solutions. The library supports features such as offline maps; lines and polygons overlays supplement to point markers overlays available in Google Maps API, KML overlays, routing service, etc (Nutiteq, 2011a). The Nutiteq Maps library is distributed under the open source General Public License (GPL).

The full overview of the functionalities available in the library and the guidance can be found in the Nutiteq Maps Lib Developer section (Nutiteq, 2011b). The component comparison between the Android Nutiteq Maps Lib SDK and Google Maps library is shown in Table 4.2.

Table 4.2. Android Nutiteq Maps Lib SDK and Google Maps library components comparison table (Nutiteq, 2011c).

	Nutiteq Library	**Google Maps library**
Map sources	OpenStreetMap CloudMade Navteq (MapTP) DeCarta DigitalGlobe (satellite and aerial) BLOM Urbex (aerial) Bing (Microsoft) Maps Yahoo! Maps WMS API TMS API Ka-Map API *Custom map sources*	Google Maps Google Aerial images Google StreetView
Offline maps, caching	Caching in memory **Permanent caching in SD card Pre-loaded maps from SD card Pre-loaded maps from application installer package**	Caching in memory only
Projections	WGS84, Spherical Mercator Mercator *Custom projections*	WGS84 for data Spherical Mercator for maps
Map overlays	Points User dynamic location **Lines Polygons KML overlays with points, lines and polygons Raster overlays**	Points User dynamic location
Routing service	Included, using OSM-based (CloudMade, Yournavigation) or own custom servers based on OpenLS	No
Geocoding (address search) service	Included, using OSM-based (CloudMade, Namefinder) or own custom servers	No
Visual indicators	Customizable DownloadDisplay Customizable ZoomDisplay Customizable Copyright overlay	None
API Compatibility with other mobile platforms	Java ME (J2ME) RIM BlackBerry Symbian (with J2ME)	None
Licensing terms	Commercial royalty-free licenses available from Nutiteq. Free for GPL applications and developers. Commercial support available. Source code available for licensees.	Free for most applications, expensive license needed for some application types (e.g. fleet management). No official commercial support, no source code available.

CHAPTER 5. The Study of the datasets

5.1. Introduction

This chapter describes the project data preparation which was accomplished by collaborating with Cleber Arruda (Geomatics Master Student in Lund University) and the workflow was discussed and commented by supervisor Lars Harrie and employees of Lund Municipality. The two datasets of Lund municipality was described and the one were chose that fits the aim of applications further developed by me and my collaborator. The application conducted by Cleber Arruda is a development of own pedestrian route network service along with the study described in this thesis the development of Android application for pedestrian navigation in Lund City; consequently this study is conducted from the perspective of pedestrians. The two datasets of the Lund municipality area are: OSM road network data (status November 2010) and the NVDB road network dataset. The latter was received from the Municipality of Lund (status November 2010).

5.2. Study area

The study of the datasets where done for the municipality of Lund. It is located in the southernmost region of Sweden called Skåne (Figure 5.1). The municipality of Lund includes urban areas such as Lund, Dalby, Veberöd, Genarp, Södra Sandby, Toma Höllestad and Stångby. The area of the Lund municipality is approximately 443 km^2 and it is located roughly between coordinates 13° 06' and 13° 39' (E-W) and 55° 30' and 55° 48' (S-N).

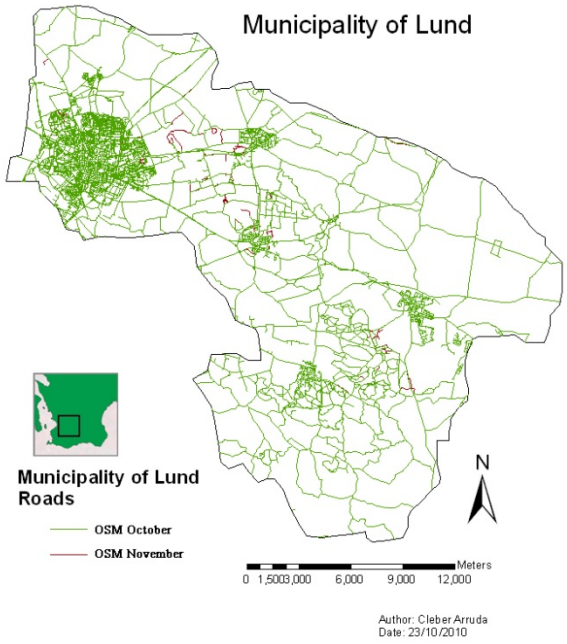


Figure 5.1. Map of Municipality of Lund (OSM road network data of October and November 2010).

5.3. Materials and Methods

Multiple steps were implemented during this study such as downloading and converting the data, installing software and tools combined with preparation and validation of the data before performing the routing calculations on it. In the Figure 5.2 the flowchart illustrates the different steps described in this chapter. The steps are as follows:

- The download of the datasets
- Conversion and projection of the datasets
- Accession and manipulation of the datasets through performing routing calculations
- Visualization of the resulting maps.

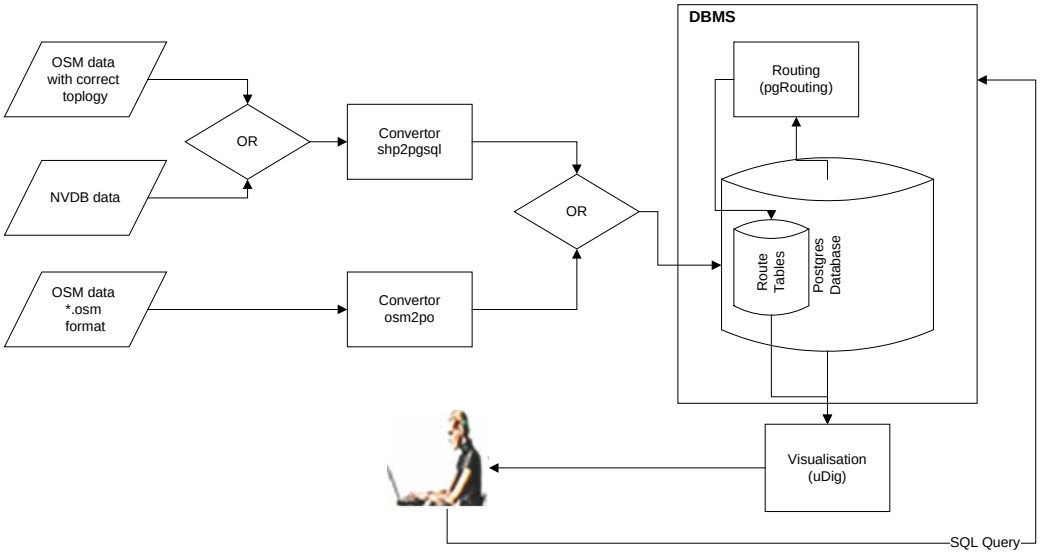


Figure 5.2. The flowchart of the methods and materials.

5.3.1. Downloading OpenStreetMap dataset

The OSM data is possible to download from the OSM Homepage (OpenStreetMap, 2010) or from OSM associate providers such as CloudeMade (CloudMade, 2010) and GEOFABRIK (GEOFABRIK, 2010). OSM road network includes not only car roads and cycle paths but also pedestrian paths for different areas in Lund municipality.

OSM Homepage provides embedded export tool for downloading the data either by filling coordinates in latitude/longitude or by use of manual drag box tool. However, the usage of this tool is limited to a specific amount of enclosed nodes which is approximately 50,000 nodes, whereas the data of Lund municipality includes more than this amount of nodes.

For downloading big areas, for example, the data of Sweden, the GEOFABRIK website can be used. GEOFABRIK website provides the road network data for large parts of world in three different file formats: bzip2 compressed XML file format (*.bzip2), ESRI Shapefile format (*.shp) and protobuf binary format (*.pbf). By assuming that all formats for the same area contain identical information, full road network of the Sweden in the Shapefile format was downloaded.

5.3.2. Spatial reference system and conversion of the datasets

A spatial reference system or coordinate reference system is a coordinate based system for uniquely identifying the location of a geographic entity on the Earth. Spatial reference system also defines map projection combined with transformation between different spatial reference systems.

The projected coordinate reference system SWEREF 99 13 30 (ESPG:3008) was used throughout the study. The projected coordinate system SWEREF 99 13 30 is a Transverse Mercator projection of the SWEREF 99 spatial reference system with the central meridian of 13°30' E, and SWEREF99 is the Swedish official realization of the European geodetic reference system ETRS89. The SWEREFF 99 13 30 is appropriate projected coordinate reference system for one zone of twelve zones for Sweden which covers the area of Skåne region where the Lund municipality is located (Lantmäteriet, 2010).

Before adding the data to the database it is required to convert and project the OSM dataset from spatial reference system WGS 84 (ESPG:4326) to SWEREF99 13 30 whereas the NVDB dataset was received in desired spatial reference system SWEREF99 13 30.

Further step was the conversion of the datasets to database format (*.sql). The OSM and NVDB Shape files conversion was done by using PostGIS's extension file called "shp2pqsql.exe". It builds database tables and holds the data in it. It is also possible to import data directly from OSM's XML (*.osm) file format into database by use of osm2pgsql tool or the OSM dataset in XML format (*.osm) can be directly convert to database file format (*.sql) using the osm2po (OSM2PO, 2010) open source convertor.

5.3.3. Description of attribute data

One of the important aspects of the datasets throughout the study is attribute data. The attributes of the data are very important while developing application based on it.

FID	Shape	OSM_ID	NAME	REF	TYPE	ONEWAY	BRIDGE	MAXSPEED
1192	Polyline	2387337	Sankt Lars vÃ¤g		unclassified	0	1	30
1193	Polyline	2387338		E22	motorway	1	0	0
1194	Polyline	2391705	BrunnshÃ¶gsvÃ¤gen		unclassified	0	0	0
1195	Polyline	2391705			cycleway	0	0	0
1196	Polyline	2399928	Sankt Petri kyrkogata		residential	1	0	0
1197	Polyline	2401871			footway	0	0	0
1198	Polyline	2401871			cycleway	0	0	0
1199	Polyline	2401879	SÃ¶dra Esplanaden		tertiary	0	0	0
...	...	...	...	...	...	...	...	...

Figure 5.3. Screenshot of OSM sample attributes data.

The attributes that OSM Shapfile includes are shown in the Figure 5.3. It includes:

- FID (Feature Identifier, type: Integer) attribute as a primary key
- Shape (type: Geometry) attribute as a geometry: for road network is Polyline
- OSM_ID (OSM Identifier, type: Integer) attribute as a foreign key
- NAME (type: String) attribute as a name
- REF (Reference, type: String) attribute: example is E22 for motorway name
- ONEWAY (type: Integer) attribute which gets value of 1 when the road is one way and 0 as a “false” value
- BRIDGE (type: Integer) attribute which gets value 1 as a “true” and 0 as a “false” value
- MAXSPEED (type: Integer) attribute which include maximum speed restriction.

From the Figure 5.3 it is apparent that not all attributes are complete, for example the MAXPEED attribute is 0 for some roads and the type “unclassified” which indicates the incompleteness of the data.

In contrast to OSM data the NVDB data is complete in this sense; it has complete attributes for the roads of the dataset. However, as it was mentioned before, the NVDB data includes only the car roads and selected cycle paths.

The attribute data of NVDB includes more complex attributes (Figure 5.4) compared to OSM data but from the side of the types of the roads it excludes the pedestrian paths and this makes it less valuable for developing the pedestrian navigation applications.

OBJECTID *	SHAPE *	NETOBJECT	FROM_DATE	TO_DATE	SEQ_NO	Nättyp	Företeelsetillkomst	ROUTE_ID *	FROM_MEASURE	TO_MEASURE	SHAPE_Length
1105	Polyline Z	{93d5f809-2f	20090204	99991231	6	cykelnät	10849:20703	10849:16256	0	1	36.393447
1106	Polyline Z	{93d5f809-2f	20090204	99991231	7	cykelnät	10849:20703	10849:16262	0	1	26.931075
1107	Polyline Z	{93d5f809-2f	20090204	99991231	8	cykelnät	10849:20703	10849:16850	0	1	25.344967
1108	Polyline Z	{93d5f809-2f	20090204	99991231	9	cykelnät	10849:20703	10849:16916	0	1	214.633506
1109	Polyline Z	{93d5f809-2f	20090204	99991231	10	cykelnät	10849:20703	10849:16922	0	1	306.376168
1110	Polyline Z	{93d5f809-2f	20090204	99991231	11	cykelnät	10849:20703	10849:17756	0	1	255.943421
1111	Polyline Z	{93d5f809-2f	20090204	99991231	12	cykelnät	10849:20703	10849:17762	0	1	154.629815
1112	Polyline Z	{3e39231c-6	20081205	20090115	0	bilnät	10850:3231	10850:3216	0	1	27.45495
1113	Polyline Z	{3e39231c-6	20081205	20090115	1	bilnät	10850:3231	10850:3220	0	1	180.040644
1114	Polyline Z	{3e39231c-6	20081205	20090115	2	bilnät	10850:3231	10850:3226	0	1	98.408922
1115	Polyline Z	{83435ae0-6	20090115	99991231	0	bilnät	10850:3231	10850:3216	0	0.33418	9.174995
1116	Polyline Z	{83435ae0-6	20090115	99991231	1	bilnät	10850:3231	10850:3216	0.33418	1	18.279956
1117	Polyline Z	{83435ae0-6	20090115	99991231	2	bilnät	10850:3231	10850:3220	0	1	180.040644

Figure 5.4. Screenshot of NVDB sample attributes data.

5.3.4. Description of geometric data

In the OSM dataset essentially the geographic entity consists of two fundamentals: tag and location. The location is related to geometric location of the entity on the Earth, whereas the tag is the property information of the entity. The location information of the OSM data is structured based on three fundamental elements in the OSM data model. The three elements are: node, way and relation. The node is point which uniquely identifies position on the Earth and can represent POIs such as a bus stop. The way element is decomposed into two sub-elements: non-closed way and closed way. Non-closed way is used to store the road data while the closed way is used to store areal data such as lakes, parks, etc. The last element is the relation that orders and relates the allied ways without characterizing the identical physical items. For instance relation can describe turn restriction (OpenStreetMap, 2010c).

In the data model of NVDB the roads are represented with two fundamentals: reference links and nodes. The reference link is described as a reference line that is the middle line of carriage way, whereas the nodes are geometrically expressed as points. The reference line is composed of multiple straight lines which are determined by the nodes (breakpoints). The rules, describing the way the nodes should be placed and describing the intersection, are strongly related to the rules forming the road network topology (NVDB, 2008). The difference between the ways of the OSM data and the reference links of the NVDB data is that NVDB reference lines are stored in 3-dimensional space in contrast to ways of OSM data that are stored in 2-dimensional space.

5.3.5. Pedestrian route calculation

The preparation step of the study is followed by the step where the data was imported to PostGIS database and by use of pgRouting functionality the route was calculated using two datasets.

PostGIS is a spatial extension of the PostgreSQL open source DBMS while the pgRouting extends PostGIS and supplements routing functionality to it. pgRouting includes several routing algorithms such as *Dijkstra, A Star* and *Shooting Star* where Dijsktra and A Star utilize vertices for calculating the route while Shooting star algorithm utilizes link to link approach for

calculation, whereas only A Star and Shooting Star algorithms exploit the heuristic approach (pgRouting, 2010).

To enable the use of pgRouting functionality the following installations were done: firstly, PostgreSQL software was downloaded and installed (PostgreSQL, 2010); secondly, open source PostGIS spatial database extension to PostgreSQL was downloaded and installed (PostGIS, 2010c), and finally, extensible open source pgRouting library was added. The compatible problem occurred with PostgreSQL version 9.0 and PostGIS, thus PostgreSQL version 8.4 was used. Additionally open source uDig software (uDig, 2010) was installed to enable spatial database representation on the desktop.

After having installed all required software, PostGIS database was created and pgRouting functionalities were added. Afterwards, the converted two datasets of Lund municipality road network data were added to our database. Because pgRouting requires several parameters as an input for shortest path algorithms, all required attributes were added for the possible algorithms: Dijkstra, A Star and Shooting Star algorithms.

To add network topology “assign_vertex_id” function was implemented which assigns source and target ID to each link. The assign_vertex_id (’<table>’, float tolerance, ’<geometry column>’, ’<gid>’) function requires 4 input parameters: first parameter is name of the table, second parameter is tolerance parameter which is binding nearby vertices depending on tolerance parameter, third parameter is the name of geometry column, and the last parameter is geometry ID. Dijkstra algorithm acquires attributes like source, target and length. A-Star algorithm acquires more attributes like latitude/longitude for start and end points for each link; and, finally Shooting Star algorithms requires additional reverse_cost attribute (pgRouting, 2010).

Since the purpose of our application is pedestrian navigation, it is necessary to tailor the data to this application. To be precise, it is essential to restrict the motorways to some degree in routing search. This can be done by use of advance routing queries. That means that the intention is not to search for the shortest path but the cheapest path and for implementing these weight cost queries were used. As OSM data includes decent amount of road types; thus, it is obligatory to assign weights to each type of the road (Example 5.1). As a result, the cheapest path calculation was working quite well for the purpose of this study (Figure 5.5).

```
UPDATE edges SET cost = 1.0;
UPDATE edges SET cost = 1.0 WHERE type IN ('tertiary');
UPDATE edges SET cost = 1.0 WHERE type IN ('primary', 'primary_link', 'secondary', 'secondary_link');
UPDATE edges SET cost = 1000.0 WHERE type IN ('motorway', 'motorway_link');
```

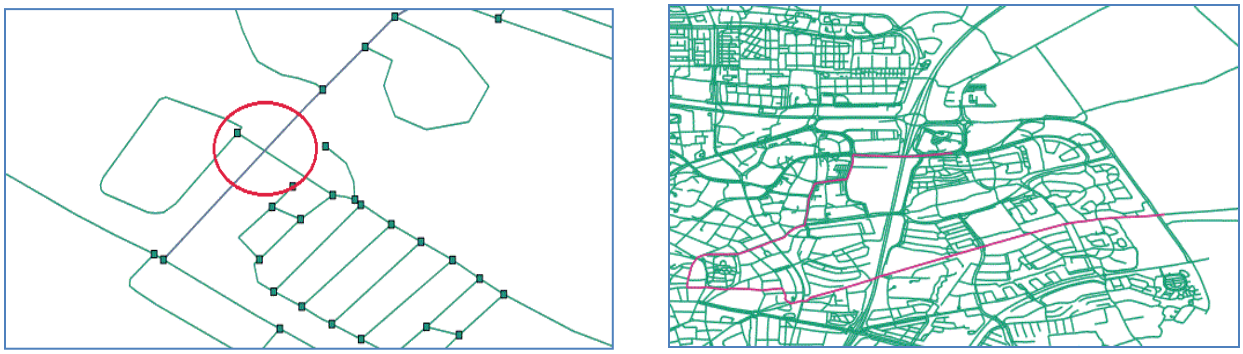
Example 5.1. Query code.

Figure 5.5. Screenshot of shortest route result with weighted cost. (NOTE: Successful routing result is illustrated, as it chooses the cheapest path by excluding motorways, and using pedestrian paths instead).

NVDB data includes only cycle roads and car roads and excludes pedestrian paths. In order to tailor the data to the purpose of the study it is necessary to reclassify all car roads and cycle paths which can be used by pedestrians. This is a huge amount of work and thus the tailoring of the data was rejected, and NVDB data in this study was used only for the purpose of comparison and analysis.

5.3.6. Missing Nodes Issue

As it was mentioned before, the OSM Shape file from the GEOFARIK was firstly used for the evaluation and it appears that Shape files from the GEOBARIK's website contains some topological problems; in some parts of the data the intersection nodes were missing and this results in connectivity problem. In the network data it is allowed not to have connection node when there is an intersection between the roads where one road is a bridge, but this is not the situation on Figure 5.6a. Here the parking lot paths were intersecting the road and it is not possible to navigate to the parking lot as the connectivity issue persists. And, as a result of this type of query, the DBMS returns 0 line segments as a routing result or it returns just a possible path but not the shortest one (Figure 5.6b).

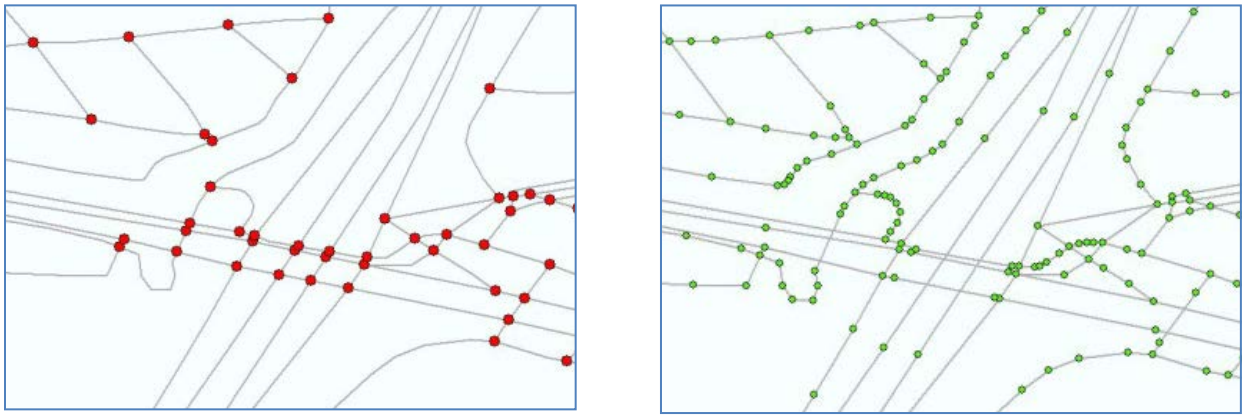


a. b.

Figure 5.6. Screenshots of missing intersection node issue.

Solution

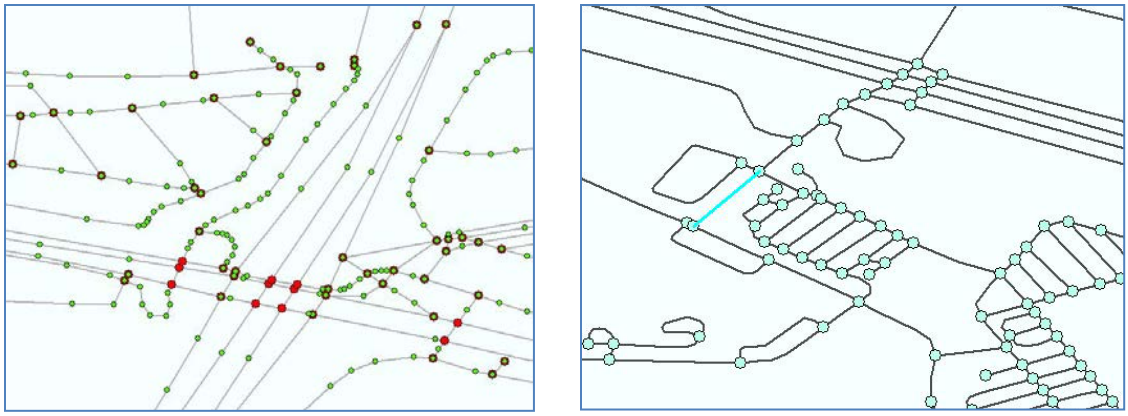
The first and simple solution of this problem is to use the OSM data in XML (*.osm) format, but still it is possible to tailor the Shape file data using ArcGIS and FME software and all nodes digitized by the users (Figure 5.7b) to build correct topology.



a. b.

Figure 5.7. a. Screenshot of all intersection nodes extracted from the road network, b. Screenshot of all the nodes digitized by the volunteers.

To perform this task the first step is to extract all intersection nodes from the road network (Figure 5.7a) which includes also unnecessary intersection nodes. That is where the bridges exist. To find out the necessary nodes to validate the topology (to split the lines in those nodes), the nodes which occurs in both Figure 5.7a and Figure 5.7b datasets should be extracted. Hence, it is necessary to overlay these two datasets (Figure 5.8a). These nodes can be used later on to split the lines of the road network which would add the missing nodes and would validate the topology. In the Figure 5.8b it is shown the validated road network shown in the Figure 5.7a.



a. b.

Figure 5.8. a. Screenshot of all intersection nodes and digitized nodes overlay, b. Screenshot of the data with valid topology.

5.3.7. Parallel link issue

A parallel links are two links which have the same source and target. The parallel link issue in this study is worth to consider, as it is limitation of the Dijkstra algorithm. While analyzing the OSM data by Dijsktra route calculation, it appears that in some parts there are two paths with same target and source, and the Dijsktra algorithm does not consider the shortest link. As it is illustrated in the Figure 5.9 the algorithm chooses the arbitrary path but not the shortest.

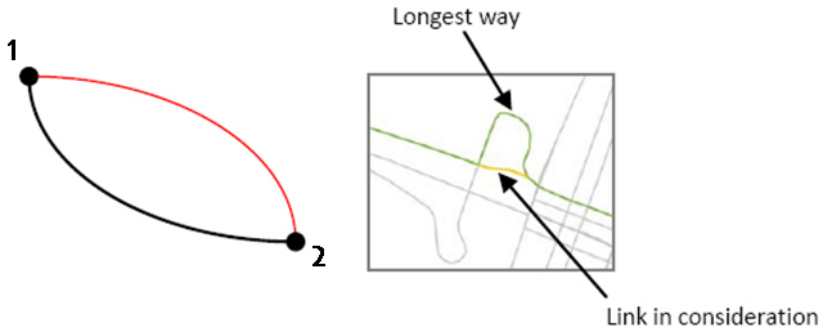


Figure 5.9. Parallel link issue.

Solution

As it was mentioned before, parallel link issue occurs due to limitation of the Dijkstra algorithm; therefore, the solution of this issue is the implementation of shooting star algorithm which considers parallel link issue and chooses the shortest link in contrast to Dijkstra algorithm.

5.4. Discussion

The study of the OSM and NVDB datasets made evident the advantages and disadvantages of both datasets in terms of data quality parameters and utility in upcoming applications (Table 5.1). In terms of logical consistency, the NVDB has defined strict rules for information model and the data obeys the information model, whereas the rules of OSM data are not so strict and not all volunteers are aware of it, which greatly influences the quality of the OSM data.

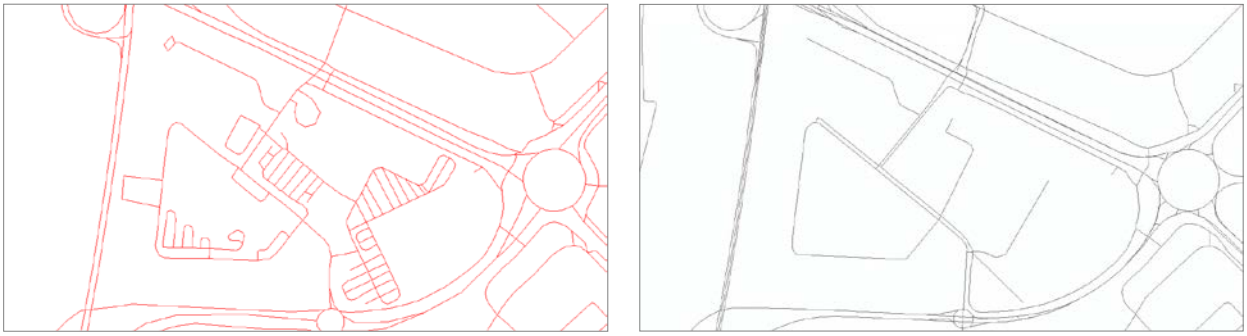


Figure 5.10. Screenshots of OSM (left) and NVDB (right) data from the same area in Lund.

The main advantage of the OSM data compared to NVDB data is that it includes pedestrian paths in addition to car roads and cycle paths (Figure 5.10). Certainly, it includes pedestrian paths partly but, because OSM road network data development is an ongoing process (Figure 5.1), better and full pedestrian paths are expected in the future. Moreover OSM is an open source project and the data can be obtained and shared without any restrictions.

Taking into consideration all the factors, we believe that utilizing the OSM data in our applications would be an appropriate choice.

CHAPTER 6. Mobile Application Implementation

6.1. Introduction

The last study of this thesis involves the development of the Android mobile map application for pedestrian navigation in the city of Lund. The purpose of the "MasterOSM" application is to provide standard navigation tools for pedestrians, which further would be used in Haptimap's location based demonstrator. The Lund Challenge demonstrator will include not only present POIs but also historical POIs, thus the standard navigation tools are essential. As it was mentioned before the demonstrator is intended not only for general users but also for visually impaired users therefore more advanced navigation tools are necessary, which is the Lund Challenge demonstrator's future development step. The task was accomplished by supervision of Miguel Molina from Faculty of Engineering (Lunds Tekniska Högskola (LTH)). The Android application was developed locally as a map application with standard navigation tools, which will be used to complement the demonstrator in the future.

6.2. Lund Challenge demonstrator-current status

6.2.1. Introduction to Lund Challenge demonstrator

The Lund Challenge demonstrator, which is the same as the Lund Time Machine Android application aims at developing tourist mobile application to make the historical and actual sights of Lund more accessible. The demonstrator is developed for Android 2.2 platform by using HaptiMap toolkit. It will assist tourists in their city explorations. Since the survey results indicate that the information on the historical sites of the city is considered to be important for the tourists; restaurants, churches and other POIs will be included in the demonstrator. Besides common users the application is intended for the visually impaired and elderly users. The demonstrator is decided to be a location based educational and recreational application that contains game-like components and also includes navigation components such as way-finding and overview that will assist users during the city exploration (HaptiMap First Prototypes, 2010). Nevertheless, it will not force the user to play the game; it could be used only for guidance in the city (HaptiMap, 2010). The Lund demonstrator includes modules such as (cf. HaptiMap, 2010):

- Creating trails/challenges
- Overview
- Navigation
- Exploration
- Mini-games and tools
- Team
- Profiling

The Creating trails module allows the user to add points and assign challenges, information and media to each point of the trial. The Overview module will provide visual and non-visual basic map information to users; that is the user should get the information by sliding the finger over the screen. The module should include current date map and an historical map in it. Additionally the radar is planned to include in the module to explore the historical content. The Navigation module will assist the user in getting the right direction towards the desired destination. The Exploration module will allow the exploration in any direction and getting information about the object of interest. The Mini-games module will include the games such as to find the locations or to solve the puzzles (quizzes). The Team module will allow specifying team members and finding the location of the member. The Profiling module will allow the user to personalize the application to own preferences (HaptiMap, 2010).

6.2.2. Lund Time Machine - current status

The summery of the components of the Lund Time Machine first demonstrator are presented below (cf. HaptiMap First Prototypes, 2010).

•The application allows following the pre-designed trails stored in a gpx file format using HaptiMap xml schema, which makes easy sharing of the trials possible.
•It allows following the trial based on pointing and scanning where the mobile device is giving sound and/or vibration feedback.
•The tourist oriented content of the application is focused on historical content: text, pictures, written and oral descriptions, and historical sounds.
•The activities in this application are related to historical sights (in this version it is a quiz).
•The application is intended for the users with different visual capabilities. The application contains visual at the same time non-visual interface components. Elder and visually impaired users in this version can follow the trials and hear information about historical sites.
•The OSM data is used in this application (and later the historical maps will also be used in this application).

6.2.3. Lund Time Machine user interface

The splash screen of the application is shown on the Figure.6.1. The main menu of the application has 4 subsections (Figure 6.1) (Lund Time Machine, 2011).

1. Ny slinga

By using this choice the user can choose a new trial or start from the beginning. Upcoming screen will allow choosing the gpx file from the phone storage.

2. Fortsätt/återgå

This section allows going back to previously guided trial.

3 Utforska

This section allows the user to explore the city without playing the game. In the current state it displays the map with the marker showing the current location of the user.

4 Avsluta

The Avsluta choice is to close the application, which is preceded by the appearance of the confirmation dialog.

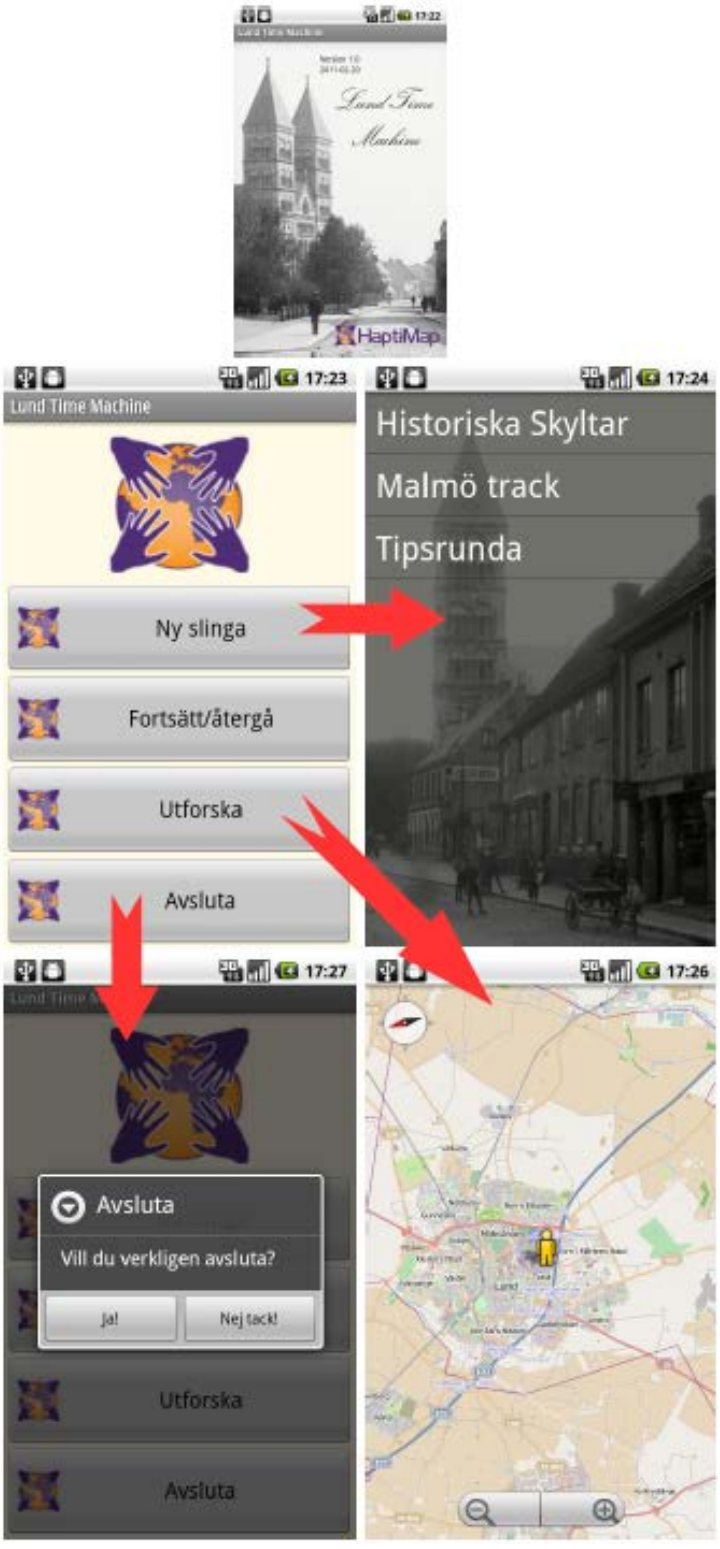


Figure 6.1. Lund Time Machine splash screen (on the top) and user interface (Lund Time Machine, 2011).

6.3. Functional Requirements (FR)

1. ***OSM Map***: The application should be able to integrate the OSM to Android environment that is to interpret OSM or CloudMade TMS tiles on the phone screen.
2. ***GPS Position:*** The application should be able to locate the position of the user on the OSM using integrated GPS connection.
3. ***Compass:*** The navigation arrow shows the direction of the user on the map.
4. ***Pedestrian Route:*** The application should be able to get from routing service the shortest pedestrian path from the current position to the destination point and overlap the route on the map.
5. ***Own POIs:*** Allows the user to choose the predefined POIs and get the route from user's position to the certain POI.
6. ***Show Direction:*** The application should assist the users by interpreting the turn signs on the map and assigning the titles to them, for ex.: "Head South on Kyrkogatan, 19m".
7. ***User Navigation:*** Allows the user to get the new route from updated user's position every 4 meters of walk.

The statuses of the FRs are shown in the Table 6.1.

Table 6.1. Functional requirement statuses (NOTE: the caption *must* represents compulsory components of the application, the *should* caption represents the optional components of the application).

FR Title	Status
FR1. ***OSM Map***	Must
FR2. ***GPS Position***	Should
FR3. ***Compass***	Should
FR4. ***Pedestrian Route***	Must
FR5. ***POIs***	Must
FR6. ***Show Direction***	Not a requirement
FR7. ***User Navigation***	Should
FR8. ***Routing in parks and open areas***	Must

6.4. MasterOSM Client-Server Architecture

As mentioned in the section 4.3 the applications generally consist of three fundamental components: data, logic and interface (presentation). In the client-server architecture the process responsibilities are partitioned between client and server, hence the thick -, medium - and thin-client terms are describing the content of process responsibilities in the client.

The three components of the MasterOSM application are:

1. Presentation (Interface) – This component is representing the user interface and enables the user interaction with the processes of the business logic component. The MasterOSM application allows the user to get the map of the surround area with the current position of the user and also allows the user to choose the POI for navigation. Furthermore the zoom buttons allows the interaction of the first tire component with the second tier business logic component.

2. Business Logic - The business logic component is linking the processes to the application logical layer. The MasterOSM application is medium-client application as the logic component is divided between the client and the server. The client is responsible for zoom-in and zoom-out actions, also for direction assistance while the server is responsible for responding the map tile in different zoom levels and routing service.

3. Data/Resource – The access to the data component is protected from the user. The data component of the application is divided between the client and server as the business logic. The positional data of the POIs is stored in the client side in the application itself, however all the map tiles and also the road network data is stored in the remote server. The access to the server data is done only through the business logic component which processes the data and returns the necessary tiles and the route to the client.

6.5. Sequence Diagram

To depict the structure of the application the sequence diagram of the application was created.
The sequence diagram illustrates the sequence of the activities taking place in the application (Figure 6.2). The user navigation process is illustrated in the steps.

- Once the user runs the application, the application starts by downloading the map tiles of the Lund City at certain zoom level, as it takes a while to get the position of the user from GPS receiver.

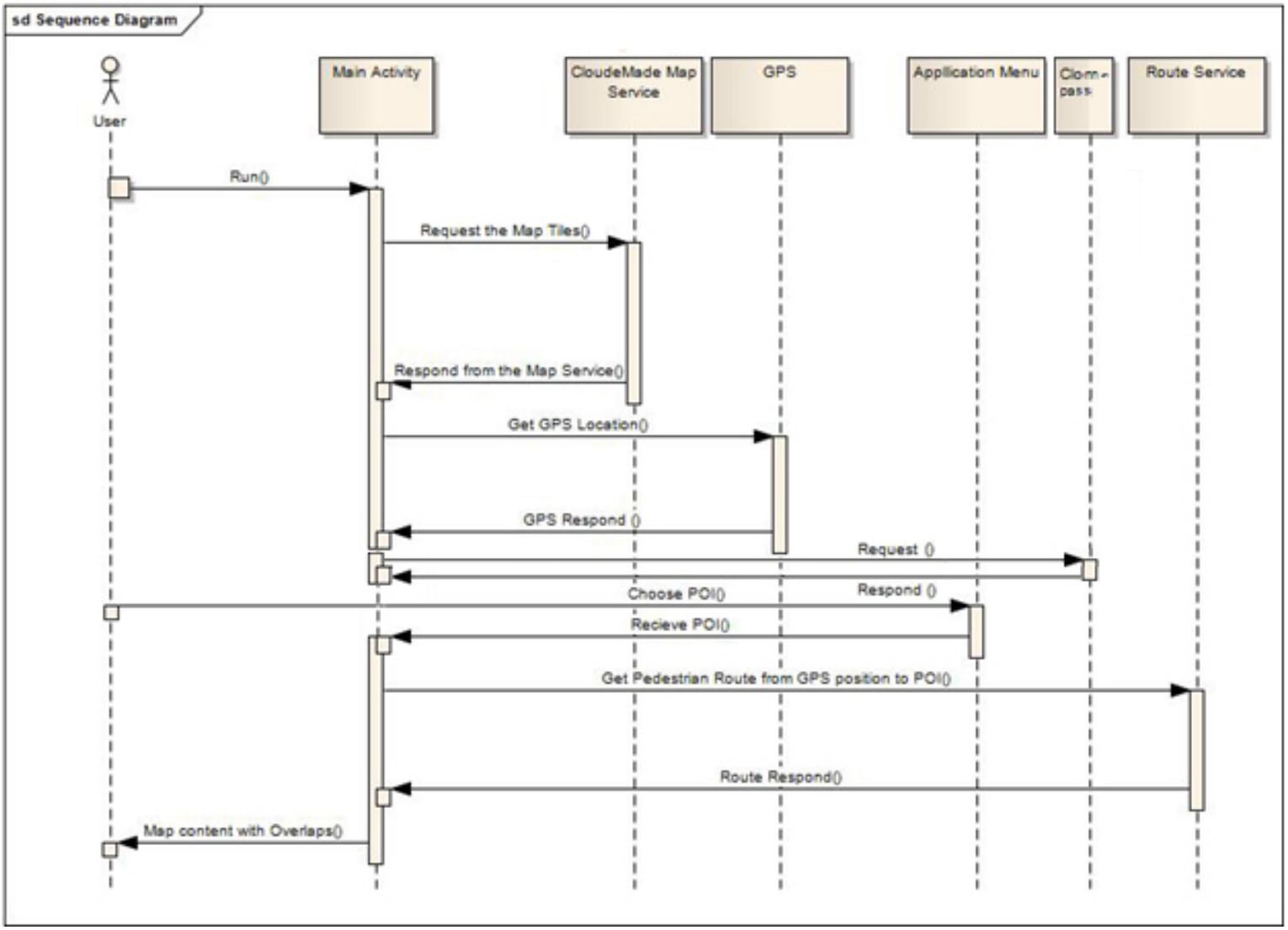


Figure 6.2. The sequence diagram of the MasterOSM application.

- When the main Activity of the application receives the GPS position the application shows the user position on top of the map. This is the default level of the application. The application menu contains two menu buttons: the POIs button and the Exit button.
- As the application gets the GPS coordinates of the user and the orientation information of the device from the Orientation sensor, it enables the navigation arrow, which depicts the current position of the user on the map and the relative direction on the map.
- The POIs button activates the List Menu which allows choosing one of the predefined POIs.
- When the POI is chosen the application sends the HTTP request with the current position and the POI as a destination point and receives the route from the Routing Service. Later it draws the route on the map with turn directions and when the user taps the direction signs the tool tip appears with the assistance message such as “Turn right at Stortorget, 39m” or “Continue 0.2km”.
- The application updates the route every 4 meters of user’s walk.
- The Exit button in the menu closes the application.

6.6. User Interface

As it was originally planned the MasterOSM application should provide the standard navigation tools for pedestrians and will be used during the development of the Lund demonstrator. The application was developed as a single application which includes the base OSM map from the CloudMade complemented with standard navigation tools.

The MasterOSM application assists the user in several ways, first it allows the user to indentify the current position of the user on the map combined with the direction of the user on the map (Figure 6.3), secondly the application allows to choose the POI from the list menu (Figure 6.4) and thirdly it allows the visual navigation of the pedestrians as the route is updated every 4 meter of walk (Figure 6.5) and at last the tool tips can be used in future for voice navigation; assisting visually impaired users.

The application user interface prototypes were designed considering the Android standard design guidelines.

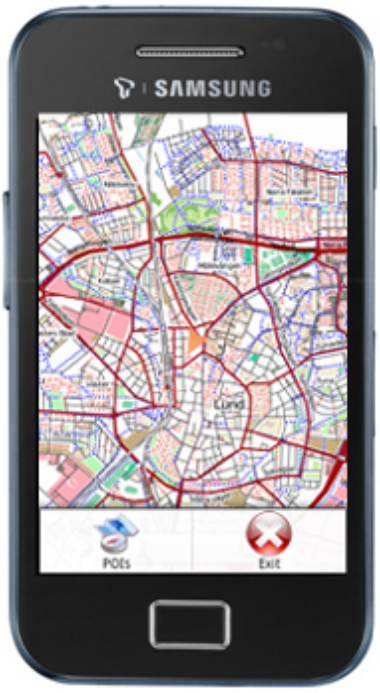


Figure 6.3. Application Main Screen

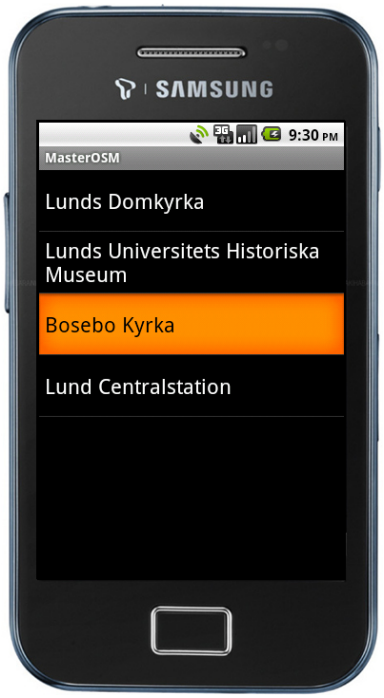


Figure 6.4. POIs Menu

a. b. c.

Figure 6.5. Navigating to POI combined with tool tip assistance
(NOTE: The route is being updated every 4 meters of walk)

6.7. Implementation details

The Android application development were done using Eclipse open source development platform, complemented with ADT[1] extension. The Android fundamentals and all the information concerning the development were obtained from the Android Developers homepage (Android Developers, 2011d).

Before getting into Nutiteq library the basics of the Android platform and application development process where studied in some depth. All the components of the Android is described in the Android Developers homepage (Android Developers, 2011d); tutorials concerning different parts and components were also in interest as sometimes it is hard to be guided only with class description. Various tutorials regarding the Android components are available in section of the Resource in the Homepage; examples are subclasses of the View class: ListView, MapView, etc. The ListView was used in this application, it allows to choose the POI to be navigated. The MapView from Google API were also tried and as it was mention before the MapView from Google API does not allow integrating tiles from others sources than Google Maps and also it does not allow to use routing services and to have a line or polygon overlays.

For understanding the components of the Nutiteq library, firstly the basic components of the library was investigated from the Nutiteq Android mapping tutorial where it starts by describing

the use of the BasicMapComponent and the properties and methods of it (Nutiteq, 2011c). The BasicMapComponents allows the integration of map tiles from all the sources mentioned in the section 4.6. The information and sample code of how to use the routing service using Nutiteq library is available at Nutiteq website's forum. It helps easy to get into Nutiteq routing components and after to adapt it to the purpose of own application.

The Android GPS and Orientation sensors where also used in this application. To understand how to use these sensors, how to get the sensor data using different listeners, the Android developer documentation was used from the Android developers website (Android Developers, 2011d).

The application was run first on the Android emulator included in Android SDK; the GPS receiver has been also simulated. As application was using data from Orientation sensor, it was necessary to use simulator to simulate the sensor, for that reason the Sensor Simulator from *Openintents* open source project was used (Sensor Simulator, 2011) (Figure 6.6) which allows the real time sensor data simulation. Three values of the Orientation sensor are *yaw*, *pitch* and *roll*. The first value yaw is simulating the compass in degrees and the range is from 0 to 360, where 0 (360) is representing the North direction, and this value was used in the application. Moreover, the pitch is the tilt of the top of device and roll is the side to side tilt of the device where the ranges of the values are -90 to 90 and -180 to 180 respectively (Sensor Simulator, 2011).

The application was build for Android SDK version 2.2 and testing was done on the Samsung Galaxy Ace phone with Android OS version 2.2.

6.8. Feasibility study of navigation in open areas

From all the required components of the application only the routing in open areas where not implemented as there is no already implemented routing service which implements the routing in open areas. However the routing in open areas is feasible whereas it is necessary to implement a pedestrian routing service on improved OSM data. To improve the road network data from OSM it is necessary to add all the possible paths in open areas, an example is shown on the Figure 6.6, where on the Figure 6.7a is shown the original road network data while on the Figure 6.7b it is shown the same open area with already added all possible paths which will enable the routing in the open area.

The study is performed in PostgreSQL database with enabled PostGIS spatial extension. To add the path in PostGIS it is necessary to create the line with two points extracted form already existing paths, and to create the line the ST_MakeLine(Point_source, Point_target) method were used which returns the LineString.

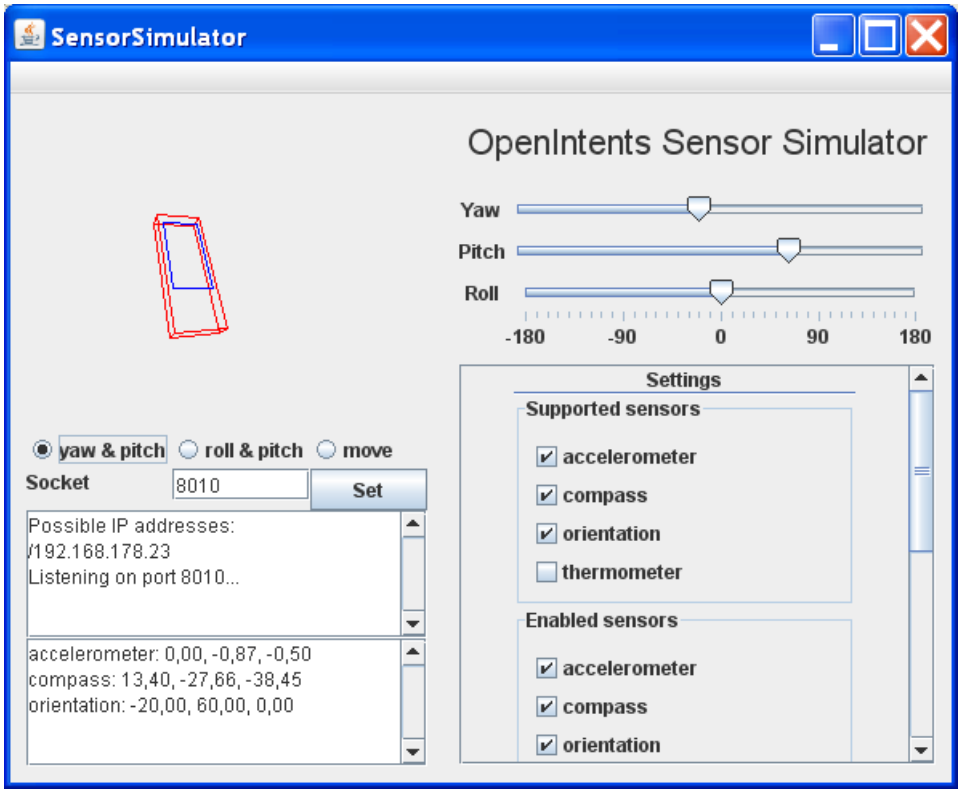


Figure 6.6. Sensor Simulator UI (Sensor Simulator, 2011).

As most converters import the *.osm xml data to PostGIS database with MultiLineString geometry the ST_Multi() method should be used to convert the created LineString to MultiLineString before adding the new created path to the database.

a. b.

Figure 6.7. a. Screenshot of the open area and the road network from OSM, b. Screenshot of the open area with all the paths added to the road network to enable the routing in the open area.

On the Figure 6.8a is depicted the routing result calculated on the original OSM data using the Dijsktra algorithm from pgRouting, and here it can be seen that the route overpasses the open areas. To ensure that the routing is possible in open areas using upgraded data the Disjkstra routing method was also run on upgraded data and here the route is passes through the open areas (Figure 6.8b).

a. b.

Figure 6.8. a. The routing result based on original data from OSM using Dijkstra algorithm from pgRouting, b. The Dijkstra routing result based on road network data with added paths in opens.

6.9. Summary

In this study the Android application development technology and structure was studied to some depth. Furthermore the Nutiteq Maps library components were also used and studied. As one of the main ideas was to use the OSM or CloudeMade TMS tiles for the base map in the application, the Nutiteq Maps library provided a good solution. First of all it allows the integration of necessary WMSs. Secondly it has better documentation compared to other open source projects. Finally despite the integration of map sources that the library is providing it includes features that are not included in ex. Google API such as the offline maps, line and polygon overlays on top of the map, online and offline KML overlays, route service, etc (Nutiteq, 2011).

The MasterOSM application has satisfied the most predetermined requirements and even additional voluntary ones such as, GPS tracking, direction assistance, etc. However, the application implementation does not accomplished thoroughly due to the lack of time and experience working with Android SDK. The application cannot be considered as complete application nevertheless as being prototype it can be used in future development of the Lund Challenge demonstrator. In terms of future development the voice assistance is required as the Lund challenge Demonstrator is foreseen for elder and visually impaired users also.

All the predefined requirements were fulfilled despite the one that was also in interest of Lund Challenge demonstrator that is the routing in the parks and open areas. All other components

used in this application could be changed, improved to give better assistant to end user. All the technologies, libraries, components used in this project were already implemented, it was just necessary to learn and integrate the components to our purpose. Additionally the feasibility study was performed that was an option to enable the routing in open areas.

CHAPTER 7. Conclusion and Future Work

The purpose of this study was to understand the technology and structure of the Android OS and application development process which was succeeded in some depth. The standard Android SDK libraries provides the possibility to integrate the Google Maps in own application with additional functionalities such as Point overlays while the OSM map integration is only possible by using supplementary open source libraries such as the Nutiteq Maps Lib SDK for Android which was used in this study. The Nutiteq API makes possible to integrate many map sources such as: OSM, CloudMade, Navteq/MapTP, etc. (Nutiteq, 2011a). Despite the map integration the Nutiteq library supports the offline maps, line and polygon overlays on top of the map, online and offline KML overlays, route service, etc. Some of these functionalities are part of public API, nevertheless to use the Nutiteq full library it is necessary to purchase a license. The CloudMade TMS tiles of the OSM were used in MasterOSM application, and moreover the routing service was used which includes the direction assistance functionality.

Open source map API:s are also available for Android, for instance the AndNav (AndNav, 2011) or Osmdroid (Osmdroid, 2011) projects. These projects provide basic components for OSM integration and interaction however the future development of the projects is uncertain.

There are several limitations of the application. Some of them are not as significant as they are already developed in Lund Challenge demonstrator such as the voice direction assistance for visual impaired users. However limitations such as the offline maps and the routing in the parks and open areas are the subject of future development.

The CloudMade routing service provides the route using only the network data of the OSM however the important aspect of the Lund Challenge demonstrator is to provide shortest path to visually impaired users, and the routing in parks or open areas is providing the paths that bypass the parks and open areas. Thereby new pedestrian routing service should be implemented based on data with added paths in open areas and parks. To achieve this first it is necessary to create a method that automatically will add all possible paths in open areas and parks for entire dataset. Later the routing service could be developed based on modified data.

As the area of Lund city is not so large thus the offline OSM maps would be an excellent solution for the Lund Challenge demonstrator. Additionally the routing techniques can be implemented to run on locally stored modified network data and this will be good combination and the application would work independent to Internet connections.

REFERENCES

AndNav, 2011: www.andnav.org/, (accessed 28 May 2011).

Android Developers, 2011a: http://developer.android.com/guide/basics/what-is-android.html, (accessed 28 June 2011).

Android Developers, 2011b: http://developer.android.com/guide/topics/fundamentals.html, (accessed 28 June 2011).

Android Developers, 2011c: http://developer.android.com/resources/tutorials/views/hello-mapview.html, (accessed 30 June 2011).

Android Developers, 2011d: http://developer.android.com/index.html, (accessed 30 June 2011).

Android Market Wikipedia, 2011:http://en.wikipedia.org/wiki/Android_Market, (accessed 28 June 2011).

Brutzman D. and Daly L., 2007: *X3D: extensible 3D graphics for Web authors*. Morgan Kaufmann Publishers.

Butler H., Daly M.,Doyle A., Gillies S.,SchaubT. and Schmidt C., 2008: *The GeoJSON Format Specification*, http://geojson.org/geojson-spec.html, (accessed 23 September 2010).

CityGML, 2010: *OpenGIS City Geography Markup Language (CityGML) Encoding Standard v. 1.0.0*, http://www.opengeospatial.org/standards/citygml, (accessed 03 October 2010).

CloudMade, 2010: http://cloudmade.com/, (accessed 02 November 2010).

Dawidson E., 2009: *Pedestrian Navigation in Stockholm*, http://www.eadept.se/Templates/Article0.aspx?PageID=618ea136-9f01-4172-b03a-0852776c65c7, (accessed 18 October 2010).

e-Adept, 2010: *Planning : Navigating : Travelling : Reporting*, http://www.eadept.se/Templates/Article0.aspx?PageID=618ea136-9f01-4172-b03a-0852776c65c7, (accessed 18 October 2010).

GEOFABRIK, 2010: http://download.geofabrik.de/, (accessed 30 October 2010).

GeoJSON Wikipedia, 2010: http://en.wikipedia.org/wiki/GeoJSON, (accessed 23 September 2010).

Giles J., 2005: *Internet encyclopedias go head to head*, Nature, 438 (7070), pp. 900-901.

Google Maps for Mobile, 2011a: http://www.google.com/mobile/maps/, (accessed 02 May 2011)

Google Maps for Mobile, 2011b:
http://en.wikipedia.org/wiki/Google_Maps_Navigation#Google_Maps_for_Mobile, (accessed 02 May 2011)

HaptiMap, 2010: D5.1 *Selection and Specification of the demonstrator applications*.

HaptiMap First Prototypes, 2010: *D5.2 First Prototypes*.

HaptiMap Homepage, 2011: http://haptimap.org/, (accessed 01 July 2011)

Harrie L., 2008a: *Lecture Notes in GIS Algorithms.* GIS Centre and Department of Physical Geography and Ecosystem Analysis.

Harrie L., 2008b: *Lecture Notes in Standardized Web Services.* GIS Centre and Department of Physical Geography and Ecosystem Analysis.

Jia T., 2010: *Exploring Massive Volunteered Geographic Information for Geographic Knowledge Discovery*, KTH Thesis.

KML, 2010: http://code.google.com/apis/kml/, (accessed 24 September 2010).

Kresse W. and Fadaie K., 2004: *ISO Standards for Geographic Information*, Springer.

Lake R., 2010: *Introduction to GML,* http://www.w3.org/Mobile/posdep/GMLIntroduction.html, (accessed 25 September 2010).

Lantmateriet, 2010: http://www.lantmateriet.se/templates/LMV_Page.aspx?id=4219, (accessed 24 December 2010)

Lund Time Machine, 2011: User Manual – Lund Time Machine.

Meng L., Zipf A. and Reichenbacher T., 2005: *Map-based Mobile Services. Theories, Methods and Implementations*, Springer.

NAVTEQ, 2010: http://www.navteq.com/, (accessed 10 September 2010).

Nivala Annu-Maaria, 2005: *User-Centred Design in the Development of a Mobile Map Application*, Helsinki University of Technology.

NVDB, 2008: *NVDB Content - Overview (version 5.3).* Swedish Road Administration, Svd department, http://www22.vv.se/filer/NVDBContents-Overview.pdf, (accessed 10 September 2010).

Nutiteq, 2011a: http://www.nutiteq.com/android-mapping-api-sdk, (accessed 30 June 2011)

Nutiteq, 2011b: http://www.nutiteq.com/mobile-map-api-sdk-guides, (accessed 30 June 2011)

Nutiteq, 2011c: *Android Mapping Tutorial*, http://www.nutiteq.com/system/files/Hello%20Map%20tutorial%20-%20Android_1_0_2.pdf, (accessed 30 June 2011)

TMS, 2011: Tile Map Service Specification, http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification (accessed 30 August 2011).

Open Handset Alliance, 2011: http://www.openhandsetalliance.com/oha_faq.html, (accessed 28 June 2011)

OpenStreetMap, 2010a: OpenStreetMap website, www.openstreetmap.org, (accessed 10 September 2010).

OpenStreetMap, 2010b: http://wiki.openstreetmap.org/wiki/Main_Page, (accessed 10 September 2010).

OpenStreetMap, 2010c: http://wiki.openstreetmap.org/wiki/Beginners_Guide_1.3, (accessed 10 October 2010).

Osmdroid, 2011: code.google.com/p/osmdroid/, (accessed 10 May 2011).

OSM2PO, 2010: http://osm2po.de/, (accessed 10 December 2010).

Peng Z. and Tsou M., 2003: *Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks*, John Wiley & Sons.

pgRouting, 2010: *pgRouting Workshop*, http://workshop.pgrouting.org/chapters/introduction.html, (accessed 05 October 2010).

PostGIS, 2010a: http://www.postgis.org/documentation/, (accessed 02 October 2010).

PostGIS, 2010b: http://www.refractions.net/products/postgis/, (accessed 02 October 2010).

PostGIS, 2010c: http://postgis.refractions.net/download/, (accessed 20 September 2010).

PostgreSQL, 2010: http://www.postgresql.org/download/, (accessed 20 September 2010).

Rusty Harold E. and Scott Means W., 2001: *XML in a Nutshell*. *A Desktop Quick Reference*, O'Reilly Media.

Sensor Simulator, 2011: http://code.google.com/p/openintents/wiki/SensorSimulator, (accessed 15 July 2011).

Shekhar S. and Chawla S., 2003: *Spatial Database: A Tour*. Pearson Education Inc., New Jersey.

TomTom, 2011: http://corporate.tomtom.com/mission.cfm, (accessed 03 April 2011).

Turton I. and Dutton J., 2009: College of Earth and Mineral Sciences, The Pennsylvania State University, https://www.e-education.psu.edu/geog585/l1.html, (accessed 21 September 2010).

uDig, 2010: http://udig.refractions.net/, (accessed 20 September 2010).

Web 3D Consortium, 2010: http://www.web3d.org/, (accessed 26 September 2010).

WMS, 2011: *OpenGIS Web Map Server Implementation Specification*, http://www.opengeospatial.org/standards/wms#overview, (accessed 15 May 2011).

WPS, 2011: *OpenGIS Web Processing Service*, http://www.opengeospatial.org/standards/wps, (accessed 15 May 2011).

Worboys M. and Duckham M., 2004: *GIS A Computing Perspective (2nd Edition).* CRC Press.

Zielstra D. and Zipf A., 2010: *Quantitative Studies on the Data Quality of OpenStreetMap in Germany*. Proceedings of GIScience 2010.

Institutionen för naturgeografi och ekosystemvetenskap, Lunds Universitet.

Student examensarbete (Seminarieuppsatser). Uppsatserna finns tillgängliga på institutionens geobibliotek, Sölvegatan 12, 223 62 LUND. Serien startade 1985. Hela listan och själva uppsatserna är även tillgängliga på LUP student papers (www.nateko.lu.se/masterthesis) och via Geobiblioteket (www.geobib.lu.se)

The student thesis reports are available at the Geo-Library, Department of Physical Geography and Ecosystem Science, University of Lund, Sölvegatan 12, S-223 62 Lund, Sweden. Report series started 1985. The complete list and electronic versions are also electronic available at the LUP student papers (www.nateko.lu.se/masterthesis) and through the Geo-library (www.geobib.lu.se)

199 Herbert Mbufong Njuabe (2011): Subarctic Peatlands in a Changing Climate: Greenhouse gas response to experimentally increased snow cover

200 Naemi Gunlycke & Anja Tuomaala (2011): Detecting forest degradation in Marakwet district, Kenya, using remote sensing and GIS

201 Nzung Seraphine Ebang (2011): How was the carbon balance of Europe affected by the summer 2003 heat wave? A study based on the use of a Dynamic Global Vegetation Model; LPJ-GUESS

202 Per-Ola Olsson (2011): Cartography in Internet-based view services – methods to improve cartography when geographic data from several sources are combined

203 Kristoffer Mattisson (2011): Modelling noise exposure from roads – a case study in Burlövs municipality

204 Erik Ahlberg (2011): BVOC emissions from a subarctic Mountain birch: Analysis of short-term chamber measurements.

205 Wilbert Timiza (2011): Climate variability and satellite – observed vegetation responses in Tanzania.

206 Louise Svensson (2011): The ethanol industry - impact on land use and biodiversity. A case study of São Paulo State in Brazil.

207 Fredrik Fredén (2011): Impacts of dams on lowland agriculture in the Mekong river catchment.

208 Johanna Hjärpe (2011): Kartläggning av kväve i vatten i LKAB:s verksamhet i Malmberget år 2011 och kvävets betydelse i akvatiska ekosystem ur ett lokalt och ett globalt perspektiv

209 Oskar Löfgren (2011): Increase of tree abundance between 1960 and 2009 in the treeline of Luongastunturi in the northern Swedish Scandes

210 Izabella Rosengren (2011): Land degradation in the Ovitoto region of Namibia: what are the local causes and consequences and how do we avoid them?

211 Irina Popova (2011): Agroforestry och dess påverkan på den biofysiska miljön i Afrika.

212 Emilie Walsund (2011): Food Security and Food Sufficiency in Ethiopia and Eastern Africa.

213 Martin Bernhardson (2011): Jökulhlaups: Their Associated Landforms and Landscape Impacts.

214 Michel Tholin (2011): Weather induced variations in raptor migration; A study of raptor migration during one autumn season in Kazbegi, Georgia, 2010

215 Amelie Lindgren (2011) The Effect of Natural Disturbances on the Carbon Balance of Boreal Forests.

216 Klara Århem (2011): Environmental consequences of the palm oil industry in Malaysia.

217 Ana Maria Yáñez Serrano (2011) Within-Canopy Sesquiterpene Ozonolysis in Amazonia

218 Edward Kashava Kuliwoye (2011) Flood Hazard Assessment by means of Remote Sensing and Spatial analyses in the Cuvelai Basin Case Study Ohangwena Region – Northern Namibia

219 Julia Olsson (2011) GIS-baserad metod för etablering av centraliserade biogasanläggningar baserad på husdjursgödsel.

220 Florian Sallaba (2011) The potential of support vector machine classification of land use and land cover using seasonality from MODIS satellite data

221 Salem Beyene Ghezahai (2011) Assessing vegetation changes for parts of the Sudan and Chad during 2000-2010 using time series analysis of MODIS-NDVI

222 Bahzad Khaled (2011) Spatial heterogeneity of soil CO2 efflux at ADVEX site Norunda in Sweden

223 Emmy Axelsson (2011) Spatiotemporal variation of carbon stocks and fluxes at a clear-cut area in central Sweden

224 Eduard Mikayelyan (2011) Developing Android Mobile Map Application with Standard Navigation Tools for Pedestrians