

Thesis in geographical information technics nr: 6

Design and Development of a Mobile GIS Application for Municipal Field Work

Gustav Ekstedt
Torkel Endoff

Program in surveying and land management
Faculty of Engineering

Department of Physical Geography and Ecosystem Science
Lund University





LUNDS TEKNISKA HÖGSKOLA

EXTM05

**Design and Development of a Mobile
GIS Application for Municipal Field
Work**

Authors:

Gustav EKSTEDT
Torkel ENDOFF

Supervisors:

Lars HARRIE
Ali MANSOURIAN
Daniel NORDSTRÖM

Opponents:

Johan FAUST
Henrik HANDELAND

Examiner:

Harry LANKREIJER
David TENENBAUM

November 20, 2012

Copyright © Gustav Ekstedt & Torkel Endoff, LTH

Institutionen för naturgeografi och ekosystemvetenskaper
Lunds Universitet
Sölvegatan 12
223 62 Lund

Telefon: 046-222 30 30
Fax: 046-222 03 21
Hemsida: <http://www.nateko.lu.se>

Examensarbete i geografisk informationsteknik nr: 06
Tryckt av E-huset, 2013

Foreword

This report is the result of a master thesis project performed by Gustav Ekstedt and Torkel Endoff, as the final part of the Surveying and Land Management programme, MSc in Engineering at Lunds Tekniska Högskola. The practical work was performed during the summer of 2012 while most of the writing was done during the fall of 2012.

First of all we would like to give a big thank you to Kart- och Mätenheten at Stadsbyggnadskontoret, Örebro. A special thanks to Daniel Nordström who had a big hand in initiating this project and showed a lot of patience while supervising us during the summer. Another thanks goes to Dennis Bergström at Sweco Positions in Örebro for giving us valuable technical advice.

Finally, a thanks goes to our supervisors in Lund, Lars Harrie and Ali Mansourian. They both helped us write a thesis that we are very proud of.

November 20, 2012

Gustav Ekstedt & Torkel Endoff

Summary

Currently, Swedish municipalities generally use paper maps to conduct field work. A mobile GIS (*geographical information system*) application could be a useful tool when conducting field work and field inventory work. In this master thesis, such an application has been developed. The application was developed for the municipality in Örebro and is called *iSurvey*. When developing a mobile application, GIS applications included, it is important to think about different interaction design aspects.

The general purpose of the application is to further digitalize the working environment and increase efficiency and reduce the amount of time used for preparation of field work. The purpose is to design and develop a tool for field inventory that can replace paper maps. More specifically, the purpose is to study interaction design, perform a needs analysis which will result in a requirement specification, study a selection of application development and web mapping techniques and finally to develop, design and implement an application.

A few existing map applications were studied to investigate what features they offered and how they were designed. The topic of interaction design and interaction design for map applications was studied. The most important aspects were the usability goals, such as *effectiveness*, *efficiency*, *safety* and *learnability*. There are principles and processes for application design and development that correspond to these goals. Among others are *Shneiderman's eight golden rules* and *product lifecycle management*. When considering interaction design for map applications, the usability goals play a major role, but the general theories of interaction design can be applied.

A selection of application development and web mapping techniques were studied. This laid a foundation for the choice of techniques used in the development of *iSurvey*. A needs analysis was then conducted through interviews with end users, which later together with the review of existing map applications lead to a requirement specification. The needs analysis showed there was a need for an application like *iSurvey*, and which the most important needs were for making the application usable. These important needs were specified as main requirements, together with a collection of other desired functionality.

After the application was developed, chosen techniques and the completed application's functionality and design could be presented. The most important functionality is to view map layers from Örebro municipality's geodatabase, to be able to make geographically bound notes and query objects on the map.

The purpose of digitalizing and making the work more effective has currently, been partly fulfilled. Because it takes time to integrate new work procedures, this purpose is currently not totally fulfilled. The purpose of replacing paper maps has been partly fulfilled for some field work procedures, but not yet for all.

Sammanfattning

I svenska kommuner används för tillfället ofta papperskartor som geografiskt hjälpmedel vid fältarbete. En mobil GIS (*geografiskt informationssystem*) applikation skulle kunna vara ett användbart hjälpmedel vid sådant fältarbete och speciellt för att genomföra fältinventering. I detta examensarbete har en sådan applikation utvecklats. Applikationen har utvecklats för Örebro kommun och kallas *iSurvey*. Vid utveckling av mobila applikationer, och även för applikationer med GIS-tillämpningar, är aspekter inom interaktionsdesign viktiga att ta hänsyn till.

Det generella syftet med denna applikation är att vidare digitalisera arbetsmiljöerna samt öka effektiviteten vid förberedelser inför fältarbete. Syftet är att designa och utveckla ett fältinventeringsredskap som ska ersätta papperskartan. Syftet är mer specifikt att studera ämnet interaktionsdesign, genomföra en behovsanalys som skall resultera i en kravspecifikation, studera ett urval huvudsakliga applikations- och karttekniker, samt slutligen att utveckla, designa och implementera en applikation.

En samling tillgängliga kartapplikationer studerades för att undersöka vilka funktioner och designupplägg som används. Ämnesområdet interaktionsdesign studerades samt interaktionsdesign för kartapplikationer. Det viktigaste inom området var användbarhetsmålen, t ex olika aspekter av effektivitet (*effective, efficient*) samt säkerhet och lärlarhet. I enlighet med dessa finns utformade principer och processer/metoder för applikationsutveckling och design, däribland *Shneidermans gyllene regler samt produktlivscykeln*. Inom interaktionsdesign för kartapplikationer spelade användbarhet en stor roll, och teorierna från generell interaktionsdesign kunde till viss del integreras även för denna tillämpning.

Ett urval av tekniker för applikationsutveckling samt skapande av kartor studerades. Detta kunde läggas till grund för ett val av teknik för utvecklingen av *iSurvey*. Sedan genomfördes en behovsanalys genom intervjuer med slutanvändare, som sedan, tillsammans med marknadsundersökningen av andra applikationer, kunde läggas till grund för en kravspecifikation. Behovsanalysen visade att det fanns ett behov över huvud taget, samt vilka behov som var de absolut viktigaste för att applikationen skulle bli användbar. Dessa behov specificerades som huvudsakliga krav, tillsammans med en samling önskad funktionalitet som bildade krav av lägre prioritet.

Efter att applikationen utvecklats kunde valda tekniker presenteras samt den färdiga applikationens funktionalitet och design. De viktigaste funktionerna i applikationen var att kunna se kartlager från kommunens databas, att kunna göra lägesbundna anteckningar samt att kunna göra förfrågningar på objekten i kartan.

Syftet att effektivisera och digitalisera arbetet har till viss del uppfyllts. Eftersom det tar tid att integrera nya arbetsrutiner har detta syfte ännu inte uppfyllts helt. Att ersätta papperskartan i arbetet har uppfyllts till viss del, för vissa fältarbeten, men inte ännu för alla.

Abbreviations

ACID - Atomicity, Consistency, Isolation and Durability
Features of a well-constructed database transaction.

ADT - Abstract Data Type
A complex object that cannot be representation as a single data type, but has a well-known semantic meaning.

AJAX - Asynchronous JavaScript and XML
A programming implementation for handling asynchronous remote calls.

API -Application Programming Interface
A programming specification that allows software components to communicate with each other.

CSS - Cascading Style Sheet
Used for styling markup language documents.

DBMS - Database Management System
The software that handles database communication and operations.

DHTML - Dynamic HyperText Markup Language
Allows for web pages to be dynamic and interactive.

DMZ - Demilitarized Zone
A secure, controlled location between an internal network and remote sources.

DOM - Document Object Model
A standardized model for objects in a markup language document.

DTD - Document Type Definition
Definition of the components in a markup language document.

GIS - Geographical Information System
A software or system that stores, manipulates or visualizes geographically bound information.

GML - Geography Markup Language
An XML dialect for representing geographical objects.

GPS - Global Positioning System
A satellite positioning system created by the American military, mainly used for navigation purposes.

GUI - Graphical User Interface

The graphical viewport of an application or software that allows a user to communicate with it.

HCI - Human-Computer Interaction

The field describing interaction between humans and digital machinery.

HTML - HyperText Markup Language

The most common markup language generally used to display content on a web site.

IDE - Integrated Development Environment

A programming environment to aid programmers in software development.

ISO - International Organization of Standardization

The leading organization for collecting and creating international standards for all purposes.

KLM - Key-stroke Level Model

A method for measuring the time it takes to perform a digital task.

LAN - Local Area Network

A local network that connects computers within a limited area.

LBS - Location Based Services

Software and services that displays information relevant to certain locations.

LOC - Lines of Code

The amount of lines of code in a software implementation. A measurement of the magnitude of a project.

OGC - Open Geospatial Consortium

An organization that handles standardizations of geospatial data and software.

RDBMS - Relational Database Management System

The software that handles communication and operations for a relational database.

REAL-analysis - Relevance, Efficiency, Attitude, Learnability analysis

A surveying method for analyzing a products usability features.

SDK - Software Development Kit

Tools for developing software with a certain purpose, with a certain package or for a certain platform.

SQL - Structured Query Language

A logical language used for communication with a database.

SVG - Scaled Vector Graphics

An XML file implementation for displaying vector graphics.

TMS - Tile Map Service

A service that implements the use of pre-generated map tiles from a database.

URL - Uniform Resource Locator

A string that uniquely represents a web based location or resource.

WFS - Web Feature Service

A service for retrieving geographical objects from a database.

WFS-T - Transactional Web Feature Service

A service for writing information of geographical object to a database.

WKT - Well Known Text

A standardized form for representing a geographical feature.

WLAN - Wireless Local Area Network

A local wireless network that connects computers within a limited area.

WMS - Web Map Service

A service for viewing and querying images of geospatial information from a database.

WMTS - Web Map Tile Service

A service for viewing and querying pre-generated images of geospatial information from a database.

XML - eXtensible Markup Language

A markup language that is suitable for storing and transporting object information.

XSD - XML Schema Definition

Definition of the components and features of an XML document.

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem statement	1
1.3	Purpose	2
1.4	Method	2
1.4.1	Theoretical part	2
1.4.2	Application part	2
1.5	Limitations	3
1.6	Target group	3
1.7	Disposition	3
2	Review of some current map applications	5
2.1	Var ligger avloppet?	5
2.1.1	Relevant features	6
2.2	Nya Hemnet	6
2.2.1	Relevant features	7
2.3	Runkeeper	7
2.3.1	Relevant features	8
2.4	OpenLayers examples	9
2.4.1	OpenLayers with jQuery Mobile	9
2.4.2	Mobile drawing	9
2.4.3	Geolocation	10
2.4.4	Relevant features	11
3	Interaction design	13
3.1	Usability goals	13
3.2	The process of interaction design	15
3.2.1	Identifying needs and establishing requirements	16
3.2.2	Developing alternative designs that meet the requirements	16
3.2.3	Building interactive versions of the designs	16
3.2.4	Evaluating what is being built throughout the process and the user experience it offers	17
3.3	Disciplines	17
3.3.1	Academic disciplines - Software engineering	18
3.3.2	Interdisciplinary fields - Human-computer interaction	19
3.3.3	Design practices - Product design	19
3.4	Principles	20
3.4.1	Norman's principles	20
3.4.2	Shneiderman's eight golden rules	21
3.5	Prototyping	23
3.5.1	Low-fidelity prototyping	24
3.5.2	High-fidelity prototyping	24
3.6	User behavior	25
3.7	Evaluation methods	26

3.7.1	User-centered analysis methods	27
3.7.2	Empirical analysis methods	27
4	Advanced interaction design in mobile GIS	29
4.1	Web map design and styling	29
4.2	Static and interactive maps	30
4.3	Mobile GIS	31
4.4	Integrating usability in maps	32
4.4.1	User-centered approach	33
4.4.2	Intelligent and context-aware applications	33
4.5	Design and development of a mobile GIS application	35
5	Technical overview	37
5.1	Mobile hardware	37
5.1.1	Smartphones	37
5.1.2	Tablets	38
5.2	Application type	38
5.2.1	Native applications	38
5.2.2	Web applications	38
5.3	Software frameworks	39
5.3.1	Software development kits	39
5.3.2	Android development	39
5.3.3	iOS development	40
5.3.4	Sencha Touch and jQuery mobile	40
5.3.5	OpenLayers and Leaflet	40
5.3.6	Open source licenses	41
5.4	Web GIS standards	42
5.4.1	Open Geospatial Consortium and International Organization of Standardization	42
5.4.2	Web Map Service	42
5.4.3	Tile Map Server and Web Map Tile Service	45
5.4.4	Web Feature Service	46
5.4.5	Projection script	46
5.5	Databases and security	47
5.5.1	Relational databases	47
5.5.2	Object-relational databases	47
5.5.3	Database transaction	48
5.5.4	Firewalls	49
5.5.5	Perimeter networking	49
5.5.6	Database products	50
5.6	Programming and markup languages	50
5.6.1	HyperText Markup Language 5	50
5.6.2	JavaScript	50
5.6.3	Cascading Style Sheets	51
5.6.4	eXtensible Markup Language	51
5.6.5	ASP.NET technology	52

5.6.6	Asynchronous JavaScript and XML	52
5.6.7	Well Known Text	52
6	Needs analysis - iSurvey	53
6.1	Methods of analyzing	53
6.2	Identified needs	54
6.3	Dismissed needs	55
7	Requirement specification - iSurvey	57
7.1	Required functionality of the application	57
7.1.1	First priority - Minimum requirements	58
7.1.2	Second priority	59
7.1.3	Third priority	60
8	Technical solutions – iSurvey	63
8.1	Chosen technologies	63
8.2	Mobile hardware solution	63
8.3	General structure	64
8.3.1	General structure of the application	64
8.3.2	General structure of web communication	66
8.4	Application and software framework	68
8.5	Web communication solutions	68
8.6	Workflow of the application script	70
8.7	Design solutions	71
9	The final product - iSurvey	73
9.1	The main graphical user interface	73
9.1.1	Hardware compatibility	74
9.2	The map	74
9.3	The toolbar	74
9.3.1	Layer switcher	74
9.3.2	Geolocate	75
9.3.3	Search	75
9.3.4	Export	75
9.3.5	Cache map	75
9.3.6	Identify	76
9.3.7	Select	76
9.3.8	Write note	76
9.3.9	Help	76
9.3.10	Status field	76
9.3.11	Color switcher	76
9.3.12	Remove feature	77
9.3.13	Modify feature	77
9.3.14	Draw point	77
9.3.15	Draw line	77
9.3.16	Draw polygon	77

9.4	General functionality	78
9.5	User Manual	78
9.6	Map layers	79
9.6.1	Changing the content of the map	81
9.7	Examples of iSurvey use	81
9.7.1	Example 1 - Surveying inventory	81
9.7.2	Example 2 - Land surveying field work	82
9.7.3	Example 3 - Environmental unit field work	84
9.8	License	85
10	Discussion	87
10.1	Dismissed requirements	88
10.2	Further development	89
10.3	Future technique - augmented reality	90
11	Conclusions	92
	References	93
	Appendix A - iSurvey figure	98
	Appendix B - Swedish iSurvey manual	99

List of Figures

2.1	<i>Var ligger avloppet? 1. The map with different sewage facilities displayed. 2. Information about a sewage facility. (Avloppsappen, 2012).</i>	5
2.2	<i>Nya Hemnet. A “drawn search”, showing items for sale within the drawn area, along with a pop up of an item that was tapped on (Nya Hemnet, 2012). The application uses Apple’s integrated maps.</i>	7
2.3	<i>Runkeeper. A run that is logged in Runkeeper. The application displays the path and milestones every kilometer on the map (Runkeeper, 2012b). The application uses Apple’s integrated maps.</i>	8
2.4	<i>A simple jQuery web application that uses OpenLayers to perform operations. The example application uses OpenStreetMap layers.</i>	9
2.5	<i>An example showing a line, four points and a polygon that was drawn by the user. The example application uses OpenStreetMap layers.</i>	10
2.6	<i>The user’s position has been tracked and is displayed with the red cross. The example application uses OpenStreetMap layers.</i>	10
3.1	<i>Usability goals for an interactive design of a certain product (Preece et al., 2007).</i>	14
3.2	<i>A simple interaction design lifecycle model (Preece et al., 2007).</i>	16
3.3	<i>Interaction design examples within its three major disciplines (Preece et al., 2007).</i>	18
4.1	<i>A simplification of a complex coast line (Dundas Data Visualization, 2012).</i>	30
4.2	<i>A mobile navigation app (GoMo News 2012). An application used while driving needs to be extra interactive to deliver information to the driver while not taking away too much time that could be spent watching the road.</i>	32
4.3	<i>Factors that can be taken into consideration by a context-aware application (Nivala and Sarjakoski, 2005a).</i>	34
5.1	<i>A tile pyramid. Every square tile in a pyramid has the same amount of pixels, but presents a view of a different extent of the map (WebGL Earth, 2012).</i>	45
5.2	<i>The DMZ network is set up as a halfway between an internal network and remote sources (Techrepublic, 2012).</i>	49
8.1	<i>General structure of the application.</i>	64
8.2	<i>Web communication between all parts of the application. 1) The tablet browser sends requests to the WMS server (explained in detail in 8.5 Web communication solutions) or the TileCache database. 2) The information in the WMS server is automatically updated once per week. 3) The information in the TileCache database is updated manually approximately once a month.</i>	67
8.3	<i>Proxies. 1) The tablet asks the proxy, via AJAX, to fetch data from the WMS database, 2) The proxy asks the WMS database to fetch the requested data, 3) the WMS database fetches and sends back the requested data as XML, 4) The proxy sends the requested data to the application where it is parsed.</i>	69

9.1	<i>Graphical user interface of iSurvey.</i>	73
9.2	<i>Toolbar with numbered buttons.</i>	74
9.3	<i>Export confirmation page.</i>	75
9.4	<i>Color selection list.</i>	77
9.5	<i>User manual.</i>	79
9.6	<i>An example of drawn features to be exported to the database. The solid lines represent plan borders.</i>	82
9.7	<i>Area to be regulated originally drawn on top of the orthophoto.</i>	83
9.8	<i>Area to be re-allotted, fitted to match property borders.</i>	84
9.9	<i>Infiltration facility inventory example.</i>	85

List of Tables

5.1	<i>Mandatory parameters for WMS requests.</i>	44
9.1	<i>Layers in the layer switcher. Baskarta and Ortofoto are the two base layers. The rest of the layers are viewed on top of these two layers. . .</i>	80

1 Introduction

1.1 Background

Municipalities in Sweden are highly dependent on having informative and reliable map systems. There are a variety of desktop map applications that fulfill this purpose. The usual solution for bringing information out into the field is by printing maps on paper. Needed map layers have to be anticipated and applied to the map beforehand. Municipalities in Sweden generally have several departments and units that use printed paper maps in field work almost every day. Examples when maps are used are land surveyors having in-field meetings with an errand's affected parties, surveyors conducting in-field inventory and employees of the environmental unit inspecting sewage.

Currently there are several types of mobile devices, such as tablets, that are well suited for serving as substitutes to the printed paper maps, according to the authors. Tablets are often equipped with a GPS receiver as well as a mobile internet connection. The large screen size, while still maintaining portability, makes tablets an optimal device for field work.

A mobile field application would be very useful, which fulfills the desires of the department employees and is designed for use on a tablet. An initial idea was formed, of an application that can substitute printed paper maps in the field and act as a dynamic map, with connections to the office's map layers. This resulted in an application that has been thoroughly developed for the *Map and Surveying unit at Örebro Municipality*. The application, called *iSurvey*, and its development process are described in the greater part of this master thesis.

The application should be made available for use in many departments, at the municipality of Örebro. The level of technical knowledge and experience varies considerably between departments. Even though the application handles advanced technical functionality, it has to be easy to use and understand. The theoretical topic that describes this type of properties of an application or service is interaction design. In interaction design the main emphasize is put on having a user-centered approach. This means having a design that is easy to learn, understand and use while including the functionality that is relevant to and requested by the users.

1.2 Problem statement

Today the common way to work is to first have to print a large paper map and make notes on it in the field, then come back to the office and have to manually rewrite these notes again on the computer. The quality of the map may be ruined by bad weather or by being folded. Employees have to anticipate which layers they need to add on the map that is to be printed. Too many layers may lead to the map being unreadable, and too few layers may lead to there not being enough information in order to complete the task. If employees realize that they've forgotten to add a layer when out in the field, there is no way for them to get the missing information.

1.3 Purpose

The general purpose of the application is to drastically reduce the amount of time used for preparation of field work, and to further digitalize the working environment. It is also to be able to conduct field inventory on a mobile device, when surveying equipment is not necessary. The problems with handling many large paper maps in the field should with this application be made minimal. The general purpose is divided into more specific points:

- Conduct a needs analysis that will result in a requirement specification for a development of a mobile map application for municipal use. They should define what an application is to be used for and how it should be implemented.
- Conduct a study on interaction design and technical solutions for map applications. The review should describe the meaning of interaction design in application development and especially when developing map applications. It should also determine how such an application could function and be designed.
- Describe main technologies that are used in map application development.
- Develop and design a mobile map application that fulfills the identified needs in accordance with theory described in this thesis.

1.4 Method

The method of this thesis is described in two parts, a theoretical part and an application part.

1.4.1 Theoretical part

For the theoretical part, literature concerning interaction design and *Human-Computer Interaction (HCI)* was identified and relevant information was extracted. A literature review was performed on the broad topic of interaction design. A more in-depth study is conducted on the part of this topic that concerns map applications. Other map applications were studied as well to see what is already available. These applications were chosen with respect to relevant functionality for developing *iSurvey*.

1.4.2 Application part

Developing an application is a long process, with several steps that should be completed before moving on to the next part. During the entire application development phase, constant communication with the department at Örebro municipality was held. The first step in the development phase was to conduct a needs analysis. Municipal units, that would be able to use the application, were identified. Employees from those units were interviewed to find out what their needs for functionality were. Other applications with similar functionality were also identified and inspected to see current status of similar applications.

Based on the needs analysis a requirement specification was formulated. This specification contains well defined functional and non-functional requirements of the application that describes exactly how the application should be implemented. The final requirement specification was approved by the department, and could be used as a base for the implementation.

When the requirement specification was finished and approved, the implementation phase of the application began. The implementation had to follow the requirement specification as much as possible, to guarantee that the final product fulfilled the requirements. If changes were desired, a discussion between developers and the department was held before updating the specification.

1.5 Limitations

For the theoretical part a limit was set on the depth of the literature review. Due to the broad discipline of interaction design, the advanced part of the study was limited to fields that are of interest when developing map applications. The study of other applications is only meant to see what other applications exist and give a brief thought of their general appearance. Further testing is outside the scope of this thesis.

The practical part has been limited to only analyzing needs and implementing the application. Some part of the functionality was already generally implemented at *Örebro municipality*. For example, the geographical information used for this application is stored in current databases. These technologies are described in the theory part of this thesis. Throughout this thesis, it will be clear what has not been implemented by the authors. The main responsibility for the continuance of the development process, however, has been the authors'.

1.6 Target group

This thesis is meant to be read by people studying or working with GIS (geographical information systems). The reader should have a basic technical understanding of application development and programming languages.

1.7 Disposition

This thesis is built up by several parts. First is a motivation to this thesis and development of an application. The second part is a more theoretical part, containing a comparison of other map applications, a literature review on interaction design, and an in-depth review on interaction design in map applications. This part also contains a technical description of technologies encountered in map application development. The third and largest part of this thesis covers a description of the full development phase. In this part the development phases are described, from the first idea and needs identification, to a final resulting application. This result is then discussed and evaluated in the fourth and final part of the thesis.

2 Review of some current map applications

In this chapter three map applications and a set of examples are examined. The relevant part of the functionality for this thesis is described. The applications and examples are:

- Var ligger avloppet?
- Nya Hemnet
- Runkeeper
- OpenLayers examples.

The purpose of this review of applications and examples is to give a general idea of possible features in implementation and design of a web based map application. The applications are chosen because of their relevance to this project with respect to their design, functionality or area of use. The applications' relevant features are presented individually for each of them.

2.1 Var ligger avloppet?

Var ligger avloppet? is a mobile application that was developed for Västerås municipality for inspection of individual sewage (*Avloppsappen, 2012*) (see *Figure 2.1*). It shows a map with sewage facilities being the main focus points. Different types of facilities are displayed with different types of symbols. The application fetches information about the facilities directly from the municipal case management system, and allows for viewing and editing of the information out in the field. Adding new facilities is also possible. The application works offline. The application can also display facilities that are within a 500 meters radius from the user's current position. The application uses a map provided by Västerås municipality.

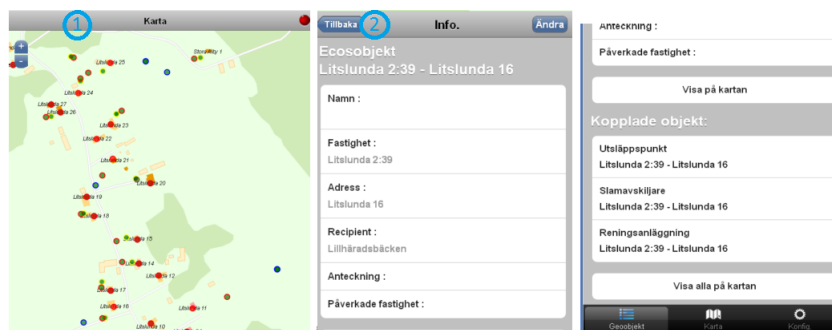


Figure 2.1: *Var ligger avloppet?* 1. The map with different sewage facilities displayed. 2. Information about a sewage facility. (*Avloppsappen, 2012*).

2.1.1 Relevant features

Features that are relevant to developing *iSurvey* are:

- Fetching municipal geographical information from a database.
- Adding features to the map.
- Writing information to a database.
- Displaying user's current position.
- Working in offline mode.

2.2 Nya Hemnet

Hemnet is one of Sweden's most popular websites for searching homes and apartments that are for sale (*Hemnet, 2012*). Their application for mobile devices, called *Nya Hemnet*, allows the user to search for items that are for sale in different ways. There is a search function where the user can type the name of the area in which she wishes to view items. There are also a number of parameters the user can decide upon, such as the price, the area of the home and key words like *balcony*. The search will display all the items that match the criteria that the user chose. There is also a tool that allows the user to draw an area on the map, and the application will display all items that are for sale within that drawn area, a "drawn search" (*see Figure 2.2*). Tapping on the items displayed on the map will display a pop up with a photo and a few pieces of relevant information. Tapping on the pop up brings you to a page with more photos and information about the item for sale. There are also several more features which allow the user to save searches and monitor items that are for sale. The application uses *Google maps*.

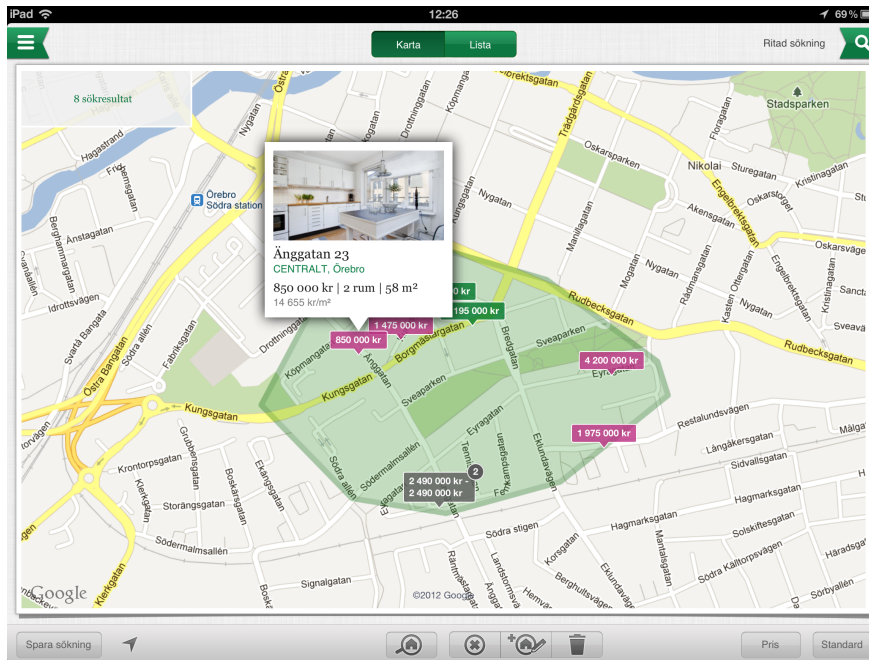


Figure 2.2: *Nya Hemnet*. A “drawn search”, showing items for sale within the drawn area, along with a pop up of an item that was tapped on (*Nya Hemnet*, 2012). The application uses Apple’s integrated maps.

2.2.1 Relevant features

Features that are relevant to developing *iSurvey* are:

- Drawing features on the map.
- Locating user’s current position.
- A search function.

2.3 Runkeeper

Runkeeper is an application for managing runs and other recreational activities via your mobile device (*Runkeeper*, 2012a). The application allows the user to track and log a run, using the device’s GPS, on a map. Other information concerning the run, such as pace, distance, calories burned and time is also logged. The path of the user is displayed on the map, along with milestones for every kilometer (see *Figure 2.3*). The application uses *Google maps*.

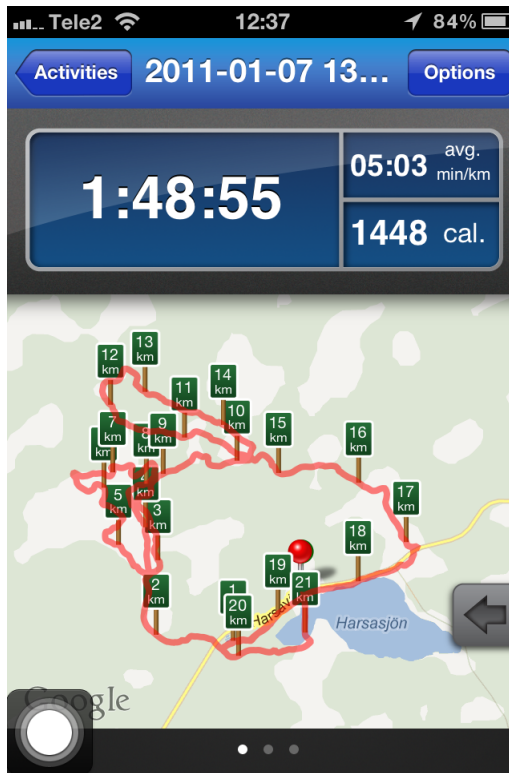


Figure 2.3: Runkeeper. A run that is logged in Runkeeper. The application displays the path and milestones every kilometer on the map (Runkeeper, 2012b). The application uses Apple's integrated maps.

2.3.1 Relevant features

Features that are relevant to developing *iSurvey* are:

- Tracking and logging the user's position on a map over time.

2.4 OpenLayers examples

OpenLayers provides developers with a large amount of examples of what developers can do with the JavaScript library (OpenLayers, 2012a). The OpenLayers JavaScript library is open source, unlike the previous native applications. The examples act as small applications that are designed to show one or two features of OpenLayers. Three examples are reviewed:

- OpenLayers with jQuery Mobile
- Mobile drawing
- Geolocation.

2.4.1 OpenLayers with jQuery Mobile

Developers can utilize OpenLayers with jQuery Mobile for making a web-based map application that has the look and feel of a native application. The example lets the user navigate around on a map, find the user's position, search for places and switch between different layers (*OpenLayers, 2012a*) (see *Figure 2.4*).

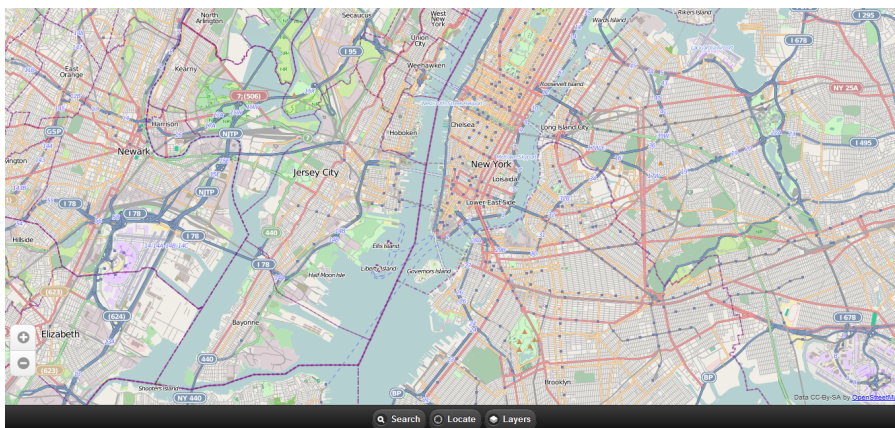


Figure 2.4: A simple jQuery web application that uses OpenLayers to perform operations. The example application uses OpenStreetMap layers.

2.4.2 Mobile drawing

A toolbar for drawing features on the map is shown on the screen. Tapping the different icons allows the user to draw points, lines and polygons on the map, move drawn features and navigate the map (see *Figure 2.5*). The usual touch gestures are used for navigating the map (*OpenLayers, 2012a*).

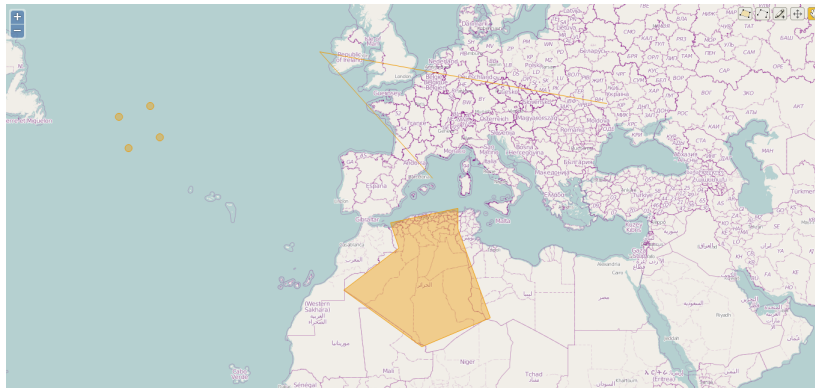


Figure 2.5: An example showing a line, four points and a polygon that was drawn by the user. The example application uses OpenStreetMap layers.

2.4.3 Geolocation

The geolocation example gives the user the option of letting the device's GPS track the user's location and center the map on it. A red cross marks the user's position and a gray circle around the cross shows the accuracy of the GPS (see Figure 2.6). Checking the "Track my position" box will track the user's position, meaning the map will follow the user's movements (OpenLayers, 2012a).

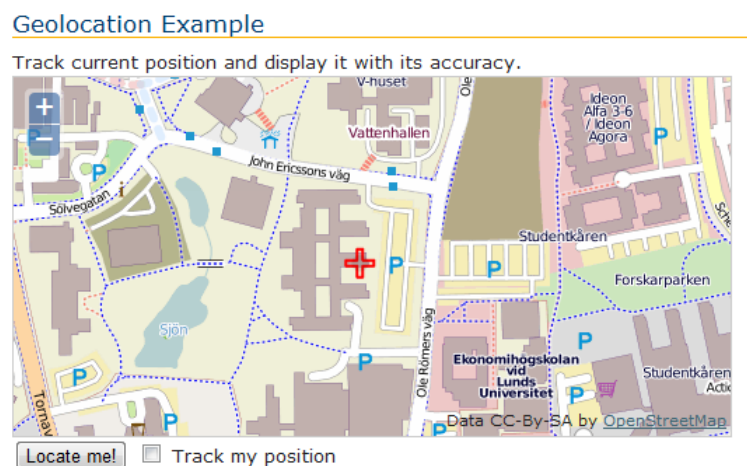


Figure 2.6: The user's position has been tracked and is displayed with the red cross. The example application uses OpenStreetMap layers.

2.4.4 Relevant features

The features relevant to developing *iSurvey* are:

- Using an open source library together with a web application framework.
- Drawing items on the map.
- Locate and display the user's position.

3 Interaction design

Interaction design is the general discipline that describes the interaction or transaction of information and actions between humans and digital products of any type. It is increasingly important along with almost all handling of machinery being built up by some sort of user interface. Such a user interface is the gateway between human will, and hardware or software performing an action. Interaction design is the theoretical topic used when *designing products*, in *human-computer interaction* or when *developing* any kind of *software*. The bottom-line focus is to make a product easy to understand and use, and not just to make it “good-looking”. Focus is also put on a product’s effectiveness to carry out a certain task while giving the user a comfortable experience.

3.1 Usability goals

The interactivity of a certain product can be defined by fulfilling several goals. These goals all revolve around the usability of the product, how interactive it is to the user (*see Figure 3.1*). The goals are (Preece et al., 2007):

- efficient
- effective
- safe
- utility
- learnability
- memorable.

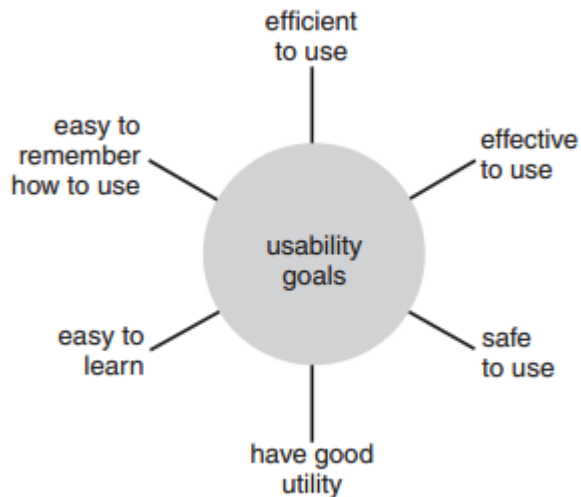


Figure 3.1: *Usability goals for an interactive design of a certain product (Preece et al., 2007).*

The *efficiency* goal is to make sure the product is efficient in performing the task it is made for, i.e. how much effort that has to be put into completing such a task. The amount of effort suitable for completing a task depends on how advanced the task is. If unlocking a door takes much more than ten seconds, a user might feel that it took much effort opening it. If ten seconds is used washing up all the dishes, the user finds it effortless.

If a product is *effective* at performing the task at hand, the result has the expected outcome. The product achieves its purpose. The air conditioning in a car is supposed to regulate the indoor climate of the vehicle according to how the driver sets the dashboard settings. If the air conditioning is successful at providing the driver with a comfortable climate when driving, it is effective.

A product that is *safe* to use is good at handling interactions without resulting in invalid states. A user that makes a minor mistake when using a product should not end up with crashed software or broken product, nor should the user risk being injured while doing so. A software application, for example, should never crash and result in work progress being lost. In the same way, a product for cooking food that is used incorrectly (in a minor way) should not result in it breaking or burning the user.

Having a good degree of *utility* might be hard to separate from being effective and efficient. An effective product can however still lack important functionality that would make it more usable when conducting a task. For example, a map application contain-

ing a search function for finding a certain object in the map might already be effective. But, by increasing the utility to also include a GPS-location function might not make the application more effective, but increases its utility.

While all the usability features presented above are fulfilled in a product, it might still be very hard to use. Learning to use it might require a user to read a manual or participate in extra education. The *learnability* of a product defines the measure of how easy it is to learn and start using. Using a new computer operating system might need a week of training or the user might be able to start using it directly without or with little extra training. If a product that is not used very often has to be retaught to a user every time it is needed, it is not *memorable*. The learnability and memorability features of a product are linked together because good learnability usually implies good memorability.

3.2 The process of interaction design

In design in general, there are three central issues that need to be taken into consideration. The first activity is understanding the requirements. The second activity is producing a design that fulfills those requirements and the third is to evaluate the design.

The core of interaction design is a *user-centered approach*. This means that not only should the technology used be the driving force behind the product, but also the real users and their goals. It is about designing for user experiences, which is defined by how a product is used by people in the real world and how it behaves (*Preece et al., 2007*). Real users should be involved through the entire development and it is important to incorporate *expectation management* and *ownership* in the users' thinking. Expectation management is making sure that the expectations of the product that the users have are kept realistic. It is better to exceed expectations than fall short. Users that have been involved and feel that they have contributed feel ownership, and are more likely to be open to the final product.

An *iterative design* is the process of designing, testing, measuring and redesigning a product. Products are redesigned based on evaluation and the feedback that is given from the testers. According to Gould and Lewis (*Preece et al., 2007*), designers never get the solution right the first time, thus making iteration inevitable. This act of focusing on the users and their goals, the use of iterative design, along with the three fundamental activities mentioned above, constitutes the essence of the interaction design process. The general flow of the interaction design process is divided into four activities (*Preece et al., 2007*):

1. Identifying needs and establishing requirements.
2. Developing alternative designs that meet those requirements.
3. Building interactive versions of the designs so that they can be communicated and assessed.

- Evaluating what is being built throughout the process and the user experience it offers.

Step four is not necessarily the last step of the process. It is something that needs to be utilized in the entire process to ensure that iteration is used. See *Figure 3.2* for a simple interaction design lifecycle model.

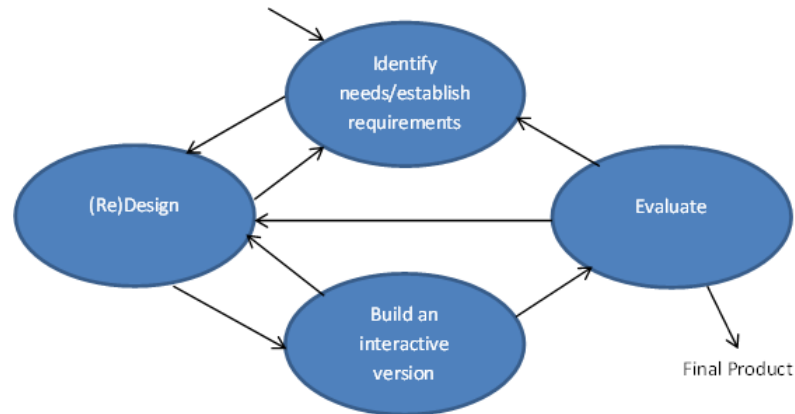


Figure 3.2: A simple interaction design lifecycle model (Preece et al., 2007).

3.2.1 Identifying needs and establishing requirements

When identifying needs and establishing the requirements for the user experience it is important to understand who the *target users* are. The needs of the users are based on their characteristics, abilities, the goals they are trying to achieve, how goals are currently achieved and if achieving their goals could be done differently in a more enjoyable manner. Needs are found through data gathering and analysis, and it is very important that users that are *representative* of their user group are consulted (Preece et al., 2007). The requirements of a product are based on the needs of the users.

3.2.2 Developing alternative designs that meet the requirements

Suggesting ideas for meeting the requirements is central to designing. This activity can be divided into two sub-activities: the *conceptual* and *physical* designs. The conceptual design of the product concerns things like what the product should do, how it should behave and what it should look like. The physical design concerns more concrete matters such as colors, sounds, images and the design of menus and icons (Preece et al., 2007). See section 3.5 *Prototyping* for further explanation of this process.

3.2.3 Building interactive versions of the designs

The easiest and most obvious way for users to evaluate the design of interactive products is to interact with them. This does not mean that a software version is required.

There are easier and cheaper ways of developing interactive versions of a design and this is done through *prototyping* (Preece et al., 2007). See section 3.5 *Prototyping* for further explanation of the various methods of prototyping.

3.2.4 Evaluating what is being built throughout the process and the user experience it offers

Evaluation of the design is important as it implies a great amount of user involvement and will lead to the final product being widely received by the target users. The process of evaluation determines the usability and the acceptability of the design or product. This is measured by how well a product fulfills the underlying requirements, how pleasing the product is to use and how many errors users make while using it (Preece et al., 2007). See section 3.7 *Evaluation methods* for further description of different methods used for evaluation.

3.3 Disciplines

There are three major disciplines where interaction design is used (Preece et al., 2007) (see Figure 3.3). The first one is *general design practices*, such as *designing products* and *graphics etc.* The second discipline concerns interaction design within *academic disciplines*, such as *ergonomics*, *engineering* and *computer science* etc. The third discipline is a collection of fields that are *interdisciplinary*, which means they are applied and integrated in one or many of the other more tangible fields. In this section these three disciplines are described along with one major field of each discipline respectively. The fields are chosen because they are either representative of its discipline or relevant to the content of this thesis. The chosen fields are *computer science/software engineering* for the academic disciplines, *human-computer interaction* for the interdisciplinary fields and *product design* for the design practices.

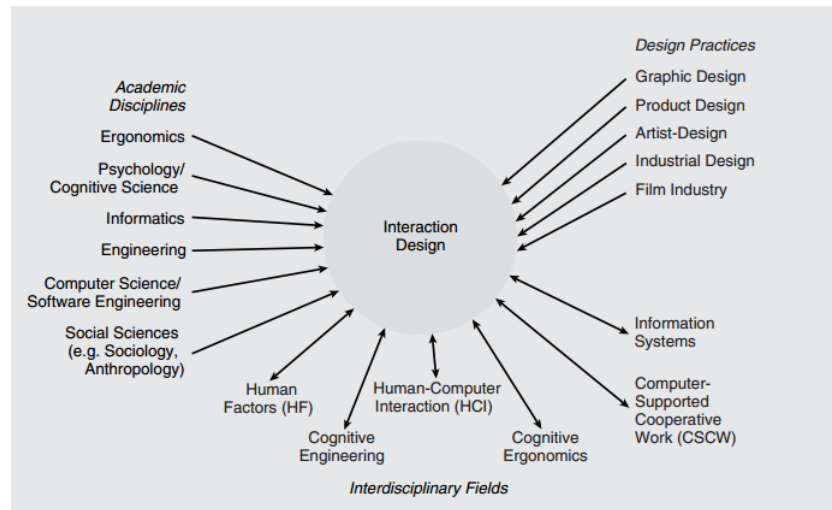


Figure 3.3: Interaction design examples within its three major disciplines (Preece et al., 2007).

3.3.1 Academic disciplines - Software engineering

Software engineering concerns the building of, usually large-scale, software systems. Software engineering has several methods that allow developers to build these systems at a reasonable *cost*, within a reasonable *time* while still maintaining a *high level of quality*. However, developers are confronted with two general problems in their projects. The first is the size of the project as higher amounts of *lines of code (LOC)* usually implies more complex systems. The second is that changes may need to be made during the development process, as new requirements can be discovered (Jalote, 2008). Some different models for handling the cost, schedule, quality, scale and change aspects of software development are shortly described below.

Software project sizes are usually measured in LOC. When developing systems that may result in tens or hundreds of thousands of LOC it is necessary to achieve high productivity by reducing cost and development time.

The International Standards Organization has formed a standard on software quality that consists of six attributes (SQA, 2001):

- functionality
- reliability
- usability
- efficiency

- maintainability
- portability.

Whereas functionality, reliability, usability and efficiency are self explanatory, maintainability and portability might not be in this context. Maintainability refers to how easy it is to maintain a functional software when faults are discovered. It can also be characterized by how readable the code is. Portability refers to how well the software can adapt and change to fit new requirements or a new environment.

3.3.2 Interdisciplinary fields - Human-computer interaction

Human-computer interaction (HCI) theory aims at improving the interaction between humans and computers through studies, planning and designing. HCI can be considered interdisciplinary because of the fact that it combines several disciplines such as computer science and human behavior science (*Helander et al., 1997*). Because of the complexity of using a computer which can do many things, unlike for example a chair which is intended only for sitting in, the interaction must be simplified to make them more usable to users.

HCI concerns techniques, methods and processes for designing, implementing, evaluating and comparing interfaces and developing descriptive and predictive models and theories of interaction. One method for this is the user-centered approach to design, mentioned in 3.2 *The process of interaction design*.

3.3.3 Design practices - Product design

Product design concerns the creation of a new product. This is done through conceptualizing and evaluating ideas which in turn will lead into a concrete product. Product designers must blend technology and design while creating a product (*Morris, 2009*). There are essentially seven steps in the product design process (*Bagnall and Koberg, 2003*):

- Analysis
 - Accept situation: The designers decide how to most efficiently solve the task at hand.
 - Analyze: The designers gather research to figure out how to solve the task.
- Concept
 - Define: The design's conditions and restraints are defined.
- Synthesis
 - Ideate: The designers brainstorm original ideas for solutions for the design.
 - Select: A few ideas are chosen and plans to make a product are made from those ideas.

- Implement: The plan is realized and prototypes are made.
- Evaluate: The product is tested and made better if needed. Complete failure may lead to starting from the ideate step again.

3.4 Principles

When creating a design for a user to interact with the designer might follow certain design principles. Principles are not equivalent to the steps in a process or model. They are more abstract features that the design of a product or interface might contain. They are there to contribute to the general feeling and overall impression that the user attains (*Preece et al., 2007*). In this section two general sets of principles for HCI are presented, *Norman's principles* and *Shneiderman's eight golden rules*.

3.4.1 Norman's principles

A set of principles for interactively designing products has been defined (*Norman, 1988*). These principles are not originally meant to cover the area of HCI and device interfaces, but can be integrated to apply to them as well. The six general principles are:

- visibility
- feedback
- constraints
- mapping
- consistency
- affordance.

Visibility The grade of visibility of a function, either on a product or some sort of interface, determines how easy it is to grasp and use (*Norman, 1988*). Usually the more important features in an interface are situated well visible to the user, while less useful or more advanced features are out-of-sight. Visual hierarchy is also used to hierarchically prioritize what buttons are more important and should be discovered by the human eye first. Size and position of icons and buttons help in making this priority.

Feedback Another important feature when performing some kind of function is that proper feedback is given to the user (*Norman, 1988*). The feedback might be a sound or feeling when pushing a button, or that a visual confirmation appears on a screen the same instant as the action is performed. Also, information has to be delivered back to the user when the performed action has a successful result. In an interface this communication has to be distributed in an appropriate way, for example in alert boxes or text fields.

Constraints It is appropriate to put constraints on functions that are not suitable for use at a certain moment (*Norman, 1988*). A product or interface can end up in a state where some functions are not usable. They should in such a case be put under constraint in some way. In computer interfaces, functions in menus are visualized as unavailable by making them non-clickable and in gray scale when the program is in a state where they cannot be used.

Mapping Mapping or natural mapping is the feature of a product or interface that resembles its functionality in the real world (*Norman, 1988*). It utilizes already known standards in everyday life to design items so that they can be interpreted by a person without prior experience with the product at hand. Standards are subtracted from physical analogies, cultural standards and biological conditions. The learnability of a product can be substantially enhanced by integrating natural mapping. Colors are useful for symbolizing certain features of a product with the help of natural mapping. As examples of this the color green usually means successful, ok or go, while red means stop, cancel or failure. Another example of mapping that is more recently developed is the fundamental touch gestures used when handling devices by touch control. Here, mapping standards are formed for zooming in or out by pinching or stretching over the screen or touchpad.

Consistency Another design principle that enhances the learnability of a product is that the design is consistent in its entirety (*Norman, 1988*). A certain design feature within a product must have the same meaning in all its instances. A button with a certain symbol cannot mean one thing in one situation while another thing in another situation. This principle should be used together with natural mapping. If for example a color is chosen to be interpreted as having one semantic meaning (red means failure), it cannot be used for another semantic meaning (red might mean warm).

Affordance How well a feature in a design signals to the user how it should be used and for what, is called its affordance (*Norman, 1988*). Much like mapping, this design principle takes advantage of what a user is expected to know without extra training. A handle can be rifled to signal to a user that it is for grabbing. It might also give other indications that signal to the user for example that it is for pushing or pulling. In an interface a button can be constructed in a three dimensional way that makes it look like a real life button, which makes the user directly aware of that it is supposed to be pushed.

3.4.2 Shneiderman's eight golden rules

Another set of principles used in interaction design is *Shneiderman's eight golden rules* (*Shneiderman et al., 2009*). These rules mainly concern user interface design, but can be used in a broader spectrum of usage areas if adjusted in an appropriate way. Although they are termed as rules they much more resemble guidelines. This means that they should work as an aid in the design process rather than a set of rules to strictly

follow. The main purpose of the eight golden rules is to increase and improve the usability of a design. The rules are (*Shneiderman et al., 2009*):

1. Strive for consistency.
2. Enable frequent users to use shortcuts.
3. Offer informative feedback.
4. Design dialogs to yield closure.
5. Offer error prevention and simple error handling.
6. Permit easy reversal of actions.
7. Support internal locus of control.
8. Reduce short-term memory load.

In this section these rules will be presented and described in greater detail.

Strive for consistency Much like the principle Norman defined, consistency is also an important property in an interface design. Sequences, commands, terminology and symbology should look and act the same all through an interface. If this property is not fulfilled, it can cause confusion for the user (*Shneiderman et al., 2009*).

Enable frequent users to use shortcuts When working with an interface regularly a user becomes an expert and should therefore be rewarded by implemented shortcuts in the interface. The purposes of these shortcuts are to reduce the amount of interactions needed between the user and the interface, and also to increase the total amount of time it takes to perform a task. Abbreviations, hidden functionality and quick routes as well as several other techniques are examples of how to fulfill the purpose of this rule (*Shneiderman et al., 2009*).

Offer informative feedback Every action performed should result in the completion of the operation along with feedback in some form. How and to what degree the feedback is communicated to the user depends on how common that type of action is. A simple press of a button might return a clicking sound or the change of the button symbol. For larger operation there might be a confirm box before initiation of the task and a progress bar showing the execution progress of the task (*Shneiderman et al., 2009*).

Design dialogs to yield closure For more complicated tasks that require several steps or operations to be performed, the interface should communicate closure to the user. Dialogs can be used to inform the user that a task has started, has reached some typical middle point and has finished. This helps the user to know what is being and has been performed. It can be seen like the interface walking a user through a more complicated task (*Shneiderman et al., 2009*).

Offer error prevention and simple error handling In designing interfaces it is very important that a faulty action by the user does not result in a serious error (*Shneiderman et al., 2009*). The implementation of the design should also offer minor error prevention and simple error handling for when a smaller error occurs. Eliminating errors in a design can be achieved by extensive testing or by using other evaluation methods (*see section 3.7 Evaluation methods*).

Permit easy reversal of actions The interface should make smaller reversals of actions possible. This is typically some type of undo function. The purpose of this property in an interface design is to make the user more at ease when interacting with it. If almost every action can be reversed to its prior state, almost no mistakes will result in major errors (*Shneiderman et al., 2009*).

Support internal locus of control The level of interactivity within a design should allow the user to feel in control of executions performed in the interface (*Shneiderman et al., 2009*). To achieve this, the user should be the one to initiate all tasks that are visibly happening in the viewport and not only serve as a responder of information. If several automatic tasks have to be performed without user interaction in between, they should be made invisible to the user to as large of an extent as possible. Instead, appropriate feedback should be used to visualize what the user is waiting for.

Reduce short-term memory load The short-term memory load put on the user should be minimized when interacting with an interface (*Shneiderman et al., 2009*). For more complicated tasks the user sometimes has to keep a vast amount of steps or items in mind at the same time, which might result in confusion for the user. For performing a single operation, although containing several sub-steps, the amount of steps should be made as few as possible. An example of where this can often be greatly reduced is when handling toolbar menu depths. It is not uncommon for there to be too many levels in a menu and its following popup windows, for the user to remember how the chain of tasks was initially started.

3.5 Prototyping

For realizing a conceptual idea in an early stage of a development process, *prototyping* is used (*Preece et al., 2007*). Prototypes are produced as *conceptual* or *physical* prototypes that can represent a vision of how a product of any kind might be designed. The purpose of prototyping is to save money and effort by producing something simpler than a final product. If the production of the final product starts and flaws are detected, it might be devastating for the product and expensive for stakeholders. An important requirement of a prototype is that it is made interactive, so that a user can actually use it in a simple but representative way. Furthermore, the purpose is to use the prototype for evaluations by both expected users and affected investors. The suitability of a product concept is tested along with its feasibility. Examinations determine if it is even possible to implement a product to fulfill the needs. Through these evaluations and tests, the requirements of the product are further developed and defined to a more real form.

Prototypes used in different stages of the design development process are divided into different types of prototyping depending on how real they look. The two major types are *low-fidelity* and *high-fidelity* prototypes, but sometimes *mid-fidelity prototyping* is used.

3.5.1 Low-fidelity prototyping

A *low-fidelity (lo-fi)* prototype is a non-realistic model of an early-stage idea of a product design. Lo-fi prototypes are often constructed in paper or cardboard. Due to its simplicity, it is cheap, simple and quick to construct this type of prototype, which means it should also be fairly easy to make quick modifications of the model. To use the prototype in an evaluating situation should also be made easy for a user. Its simplicity and unfinished state provide a low level of respect for the completeness of the product, which in turn encourages users to express ideas and criticism.

There are a number of methods for lo-fi prototyping (*Preece et al., 2007*). One prototyping method, called *sketching*, is presenting single and simple sketches. They work as informal drawings of a product or interface design. Another method, *storyboarding*, refers to a method where a predefined scenario is presented as an interactive story. Usually sketches are used for creating the steps that builds the scenario. *Prototyping with index cards* means creating slides of a more complete version of a product. The user is presented with a starting slide. When changing the state of the product (for example by clicking a link of a website), a new slide is shown to present the product's new state. One prototyping method that is used for software development is *Wizard of Oz*. The user interacts with a screen on one computer. Operations are manually controlled by a person on another computer that handles the software's intelligence, and responses are sent back to the user. This method can be compared to prototyping with index card, but in a more complex and digital form.

3.5.2 High-fidelity prototyping

When needing a prototype for representing a more realistic version of the final product, *high-fidelity (hi-fi) prototyping* is used (*Preece et al., 2007*). These prototypes are commonly used for tests and evaluations in a later stage of the development process, when requirements have been further established. Used materials are also generally more like the ones that will be used in the final product. Models in plastic, with the right feel and weight, can for example be molded to a prototype if the final product is likely to be made of plastic. For software engineering, hi-fi prototypes are often created as a simpler form of interactive software, created in an appropriate software tool.

There are however drawbacks of hi-fi prototyping (*Preece et al., 2007*). Compared to lo-fi prototyping, which is known as quick and simple, a hi-fi prototype takes a long time to develop and is usually much more expensive. When a user is shown something that looks almost finished, they tend to comment only on smaller details and not on substantial changes that might be necessary for making the product perfect. In the same way, developers are usually reluctant to change in this stage of the development

process, since much time has already been put on implementation. Expectations on the final product can also be raised to an unreasonable standard due to an attractive prototype design. The presented level of design features might not fit into a fully implemented version. Finally, the risk of a bugging prototype execution can ruin a user test by worsening the general attitude towards the product.

Although there are some drawbacks of using hi-fi prototyping, there are areas of use where it is very useful. One example of this is when promoting a new product to financial investors and upper management, to show what a project might result in. Another example is when examinations of possible technical solutions are made, to find out if a chosen technique can fulfill the needs and requirement of a product (Preece et al., 2007).

3.6 User behavior

Understanding the user's behavior requires knowledge about human *cognition*. Cognition involves cognitive processes such as thinking, remembering, learning, daydreaming, decision-making, seeing, reading, writing and talking (Preece et al., 2007). There are usually two kinds of cognition that can be brought up, *experiential cognition* and *reflective cognition*. Experiential cognition concerns user's experiences and expertise and how they allow the user to perceive, act and react effectively and effortlessly when performing tasks. Reflective cognition on the other hand requires the user to think, compare and perform decision-making when presented with a task. Reflective cognition leads to creativity and the birth of new ideas (Preece et al., 2007). Another description of cognition can be a set of processes that seldom appear alone when performing tasks (Preece et al., 2007):

- attention
- perception and recognition
- memory
- learning, reading, speaking and listening
- problem-solving, planning, reasoning and decision-making.

Of the processes described above, memory (the recalling of knowledge) is most central to interaction design.

Because humans are social and very emotive beings it is also important that the affective aspects of behavior are considered when designing software. Because of the inability of computer hardware to express emotions, the software must be designed in a way that gives the user a feeling of ease, comfort and enjoyment. It is important to reduce negative emotions and frustration when using software (Preece et al., 2007).

3.7 Evaluation methods

When working with a user-centered approach, constant *evaluation* is needed. This is to determine if the users can use the product as intended and if they have a feeling of enjoyment while using it. The product can then be modified according to *feedback* gained from the evaluation, and real problems can be fixed before the product reaches the market. What needs to be evaluated varies between different product types, but there are a few general aspects that should be taken into consideration. These include the *usability*, *aesthetics* and *emotions* of a product, as well as how well it *engages* and *motivates* the user (Preece et al., 2007). Evaluation can be done in all stages of a product's lifecycle (Eriksson, 2012):

- product concept phase
 - survey of attitude and interest
- research and understanding phase
 - focus groups
 - ethnographic studies
- requirement specification phase
 - benchmarking
 - task analysis
- design phase
 - cognitive walkthroughs
 - exploratory user tests
- implementation phase
 - heuristics analysis
 - test of validity
- marketing phase
 - beta testing
- support and further development phase
 - reports
 - logs

The evaluation can be performed in either an *empirical* or *analytical* analysis. Empirical testing refers to that some property is being tested or measured in a concrete manner, for example how long it takes to perform a certain task. If on the other hand an analytical testing is performed, focus is put on reasoning around a given property to determine what meaning it bears, for example if a product is perceived as effective or

slow. These two analysis methods result in *syntactic* and *semantic* outcomes respectively.

The evaluation can also be either *formative* or *summative*. A summative test is performed on a product to its entirety. The full scope of the product is being tested. A formative test concerns only a part of the product or some certain detail of it. An example of this is that only the speed of execution is examined in an application.

3.7.1 User-centered analysis methods

Throughout the product development and design process, a number of *user-centered analysis* methods might be used. These methods are used for determining advantages and flaws in a product in different stages of the process. A quality assessment is thus made to examine the *goals*, *users* and *context* of the product (Eriksson, 2012). The assessment is made to determine if the goals and purposes are fulfilled by this product. Also the expected user is determined along with the skills and knowledge of the user. Finally the context in which the product is to be used is determined.

In an early stage of the development process the goals, expected users and context of a product is established (Eriksson, 2012). *Brainstorming* is one way of starting this process. All thoughts about the product are gathered in a very raw form. The gathered information is later organized and a vision or idea can be articulated. Another similar process is discussions in *focus groups*. Several people are gathered, of different professional backgrounds and with different points of view. This results in that the product is critically assessed in respect to several important aspects.

Surveys are used in the evaluation process to determine the quality of a product. The quality is examined through questions about the product's *relevance*, *efficiency*, *attitude* and *learnability* (Eriksson, 2012). Such a survey is called a *REAL-analysis*. The relevance property refers to if the goals of the product are fulfilled. How well the goals are fulfilled depends on the efficiency of the product. The general attitude of the product that the user experiences, is also examined in this type of survey. Finally, how easy it is to learn to use the product is tested, through examining the learnability of it.

When a product has reached such an advanced state that it can actually be used in its proper context, evaluations can be made while it is used in its real environment. This kind of analysis, called an *ethnological study*, helps to detect flaws that are otherwise hard to identify (Eriksson, 2012).

3.7.2 Empirical analysis methods

Empirical methods are used to manage raw data and measurements in the evaluation of a product. Conclusions are later made from the results of these analyzes. *Fitts' law* and *Keystroke-level model* are examples of methods to process data in this form. The time it takes for a user to move to an object in an interface, either with a mouse or on a touch screen, is proved to be a function of the distance to and the size of the object. This

relationship is defined by Fitts' law and is used in evaluations in HCI and ergonomics (*Card et al., 1986*).

To examine the amount of time it takes to perform a series of operations, keystroke-level modeling (KLM) can be used (*Card et al., 1986*). The method is used in HCI and especially when a user interacts with a computer through a mouse or pointer. A detailed mapping of partial operation time consumption is achieved. Operations like for example going from keyboard to mouse and back takes a substantial amount of effort and time, which is discovered by using KLM. These operations can then if possible be taken out of the task.

4 Advanced interaction design in mobile GIS

Using a *geographical information system (GIS)* involves visualizing and interacting with geographical data. This is done in some kind of GIS software. A GIS needs to be able to handle representation of the geographical data, in a form that is usable for the expected users of the software. *Spatial cognition* is a term that describes how humans interact with spatial environments in real life. A GIS interface is supposed to represent reality on a screen as a digital map. Since HCI is a user-centered field, and different people have individual and varying spatially cognitive perceptions of real life, incorporating HCI into GIS interfaces is very hard.

Due to advances in current technology, GIS software has rapidly moved onto mobile platforms. Mobile devices currently include both an internet connection and GPS functionality that make them very suitable for GIS use. There are, however, substantial differences between desktop software and mobile applications, which mean higher demands must be put on interactive appliance on a mobile GIS application to lessen the cognitive load.

In this chapter interaction with GISs is examined, especially in mobile usage. Basic *map styling* techniques are examined and compared to general interaction principles. Differences between static and interactive maps are also defined. *Mobile GIS* along with *usability in maps* and a method for *designing and developing mobile GIS applications* make out the main part of this chapter.

4.1 Web map design and styling

The art of designing maps for use on the internet requires the author to deliver relevant information while indulging in simple map design. The design must be simpler than a paper map due to the restrictions the screens of mobile devices (as well as smaller computer monitors) put on the amount of information that can be viewed. Another reason for the maps to be simpler is that it will probably reach far more users if it is on the internet than if it were a paper map. This means that users with special needs or users that only use maps occasionally should be able to understand what the map conveys. It is harder to control who the end users might be, professionals or amateurs. Fewer colors should be used than on a paper map, and only the most vital information should be displayed (*Jenny et al., 2008*). Since the map is part of the user interface, the guidelines that *Norman* and *Shneiderman* present (*see section 3.4 Principles*) are taken into account.

Objects that are most important need to be chosen and be displayed graphically so that their importance is recognized. This is called *visual hierarchy* and is something the author must consider when designing the map. If an object is drastically in contrast to its surroundings, it usually signifies importance; the greater the contrast, the greater the importance of the object. There are several techniques for making maps simpler. Two widely used concepts are *generalization* and *symbolology* (*ESRI, 1996*). These features are examples of how the design principle *visibility* can be used in map

development.

Generalization is the process of removing unnecessary object detail through e.g. *selection*, *simplification* and/or *classification* (ESRI, 1996). Selection is usually done in the database through selecting only certain subsets of features to be displayed. Simplification is the process of selecting points in lines (see Figure 4.1). Most data viewed in current internet maps are simplifications of their source data. Classification is the process of grouping data with similar values or types together. To make interpretation easier for humans, no more than seven levels/categories should be used (ESRI, 1996).

Symbology is used for representing objects through pictorial symbols or representative colors (ESRI, 1996). When designing symbols the author should have *natural mapping* in mind, to use symbols that represent objects that users are faced with in everyday life. Restaurants for example should be marked with a fork and knife on the map, campgrounds with a tent, etc.

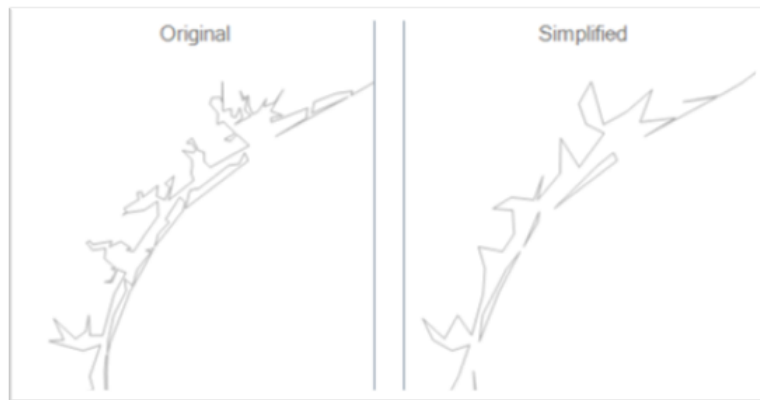


Figure 4.1: A simplification of a complex coast line (Dundas Data Visualization, 2012).

4.2 Static and interactive maps

There are two major techniques for displaying maps on the web, *static web mapping* and *interactive web mapping*. The two differ in many ways and the developer should consider which type fits the application or website the best (Arnberg and Rystedt, 2008). Static maps are basically paper maps that appear on a computer screen. They are images like *jpeg* or *gif* that are fetched and displayed. Because of their nature of being only simple images the user cannot perform any interaction with the maps at all. There is no zoom functionality, no option of adding additional layers etc. Static maps are easy to implement and require only simple HTML knowledge to incorporate in a website. Interactive maps offer the user several options for interacting with them such as zoom and pan. They are stored as more complex data structures than simple images, usually in *scaled vector graphics (SVG)* or *shapefiles*. These are more compli-

cated to implement than static maps and require *Dynamic HTML (DHTML)*, *plug-ins*, or *ActiveX* (among others) to run.

Interactive maps allow the user to do more with the map than just viewing one at a fixed scale with a fixed set of features displayed. They are however more complicated for the developer to handle and perhaps static maps are to favor if the purpose of the map is only to give the user an overall view of an area without the need to zoom or perhaps use measurement tools.

4.3 Mobile GIS

While personal computers grew dominant during the 1990s, the current era is definitely mobile based. The mobile approach on GIS uses *location based services (LBS)* which is made possible since the device is not bound to be used at home or an office. LBS are the general description of GISs that use geographical data at a certain location of interest. The GPS functionality available in most modern mobile devices along with an internet connection, make searching possible for both the current position as well as any other location of interest. There are intelligent applications available that are *aware* of the user's purpose (*see section 4.4.2 Intelligent and context-aware applications*). This is a specialized approach on using a GIS application in an "on the move" usage situation, which is of course only possible with mobile devices (*Nivala et al., 2007*).

Compared to desktop computers, there are both flaws and advantages of using mobile devices. For example, mobile devices have smaller screens, non-standardized placement of hardware keys, lesser processing power and memory volume, and they also lack standard hardware like a mouse and keyboard. They do however often use multi touch screen functionality and virtual features (e.g. virtual keyboard), which make up for some of these flaws by saving screen space and expanding available command calls. One great advantage of mobile devices is that they are mobile and that they can be used anywhere. In a situation of mobile use of a GIS, the human perceptual and cognitive senses have much more to process, than just the content shown on the screen (*Nivala et al., 2005b*). While the surroundings in a home or an office is well known to the user, many new impressions are perceived by a user standing outdoors in a forest or a public place (*see Figure 4.2*). When competing for the user's attention with real life impressions, the demands on user-friendliness are even higher (*HaptiMap, 2012*). All these features of mobile devices stress that requirements must emphasize the need for a well formed interaction design.

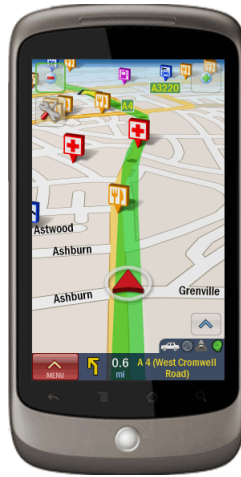


Figure 4.2: A mobile navigation app (GoMo News 2012). An application used while driving needs to be extra interactive to deliver information to the driver while not taking away too much time that could be spent watching the road.

4.4 Integrating usability in maps

As mentioned earlier maps and GISs are visualizations of human spatial cognition of the real world, or *geovisualizations*. Like for any product development, and especially for software development, integrating usability in GIS is an important phase (*HaptiMap, 2012*). Due to diversity in each user's spatial cognition of reality and how comfortable the user is when handling maps, usability aspects for map applications are hard to define (*Nivala et al., 2005b*).

A map view in a GIS can be considered part of the interface. If acknowledged as a GUI, general interaction design methods and principles can be used for designing the map as well. This approach is however not perfect. A badly designed map still contains a substantial amount of information that the user can progress and use. It might be hard to separate the most fundamental goals of usability, in a map design: *effectiveness* and *efficiency*. They both highly depend on what the purpose of the map is, its *context of use*. The different purposes that a map can have range widely from weather forecasting to property planning. A map that is badly designed for one purpose might be excellent for another. Research has found that some properties of a map can always be used as a general measurement of its quality (*Nivala et al., 2005b*). These very fundamental properties are that a map should contain up to date information, be accurate and be easily accessed. Easily accessed in this case means for example that no complex parameters should have to be known beforehand for a user to view a certain extent of the map. Instead it should only be a click away, automatically showing the extent wanted by the user (*see section 4.4.2 Intelligent and context-aware applications*).

A map usually always contains errors. Some errors are inevitable since they are based on subjective opinion of some users, for example name labeling choices of areas or buildings of interest. Errors like these are proven to be much more accepted when using an interactive, digital map rather than a paper map (Nivala *et al.*, 2007). This is mainly because of the dynamic nature of an interactive map. If a label is not shown of some building or area of interest to the user, dynamical zoom functionality can easily solve this problem. If a paper map is used, the information is either there or it's not.

4.4.1 User-centered approach

Research has shown definite benefits of user involvement and of consideration of usability aspects in GIS development (Nivala and Sarjakoski, 2005a). Along with GISs being available on mobile devices, this type of map technology is no longer used only by professional GIS users. This means applications must be made usable for both experts and amateur users, with a broad variety of the amount of map knowledge. While this makes GIS development more like software development in general, it does require changes to be made in the development process. Currently, the most common way of involving usability in the process is by using professional users and experts for evaluations. As an independent event, this is seen as a method that costs less and takes less time. Due to the changes described above though, this might result in problems further along in the development process or when the product is released on the market. In reality involving end-users from in an early stage of the process, might therefore on the contrary be more cost-effective. The greater the part is of the market that consists of amateur users, the greater the loss will be.

When involving end-users in the development and design process of a GIS, problems can occur. Much like in any software development process, where non-expert users are involved, the users don't know what they want. The developers somehow have to show the user a vision or an idea of an application, that the users can give feedback to. This is most easily done by visualizing the idea using models, mockups or prototypes. The same method can be used in GIS development, where a model is shown to the users, evaluated and improved to meet their needs (HaptiMap, 2012).

4.4.2 Intelligent and context-aware applications

A map application that can be used for a variety of purposes can be implemented to be intelligent, or context-aware (Nivala and Sarjakoski, 2005a). This means that the application, to some extent, automatically makes itself aware of the user's context of use. Through this, only chosen information can be shown in the map. It can be presented with certain themes that are appropriate for the current user. The most commonly implemented functionality that can serve as an example of this is the GPS's locating ability. By knowing where the user is, the application can suggest certain tasks to be performed. But there are also other factors of the user's context of use (HaptiMap, 2012) that affect the application and makes it adapt visualization (Nivala and Sarjakoski, 2005a) (see Figure 4.3):

- surroundings
- purpose of use
- system properties
- orientation
- location
- time
- navigation history
- social and cultural situation.

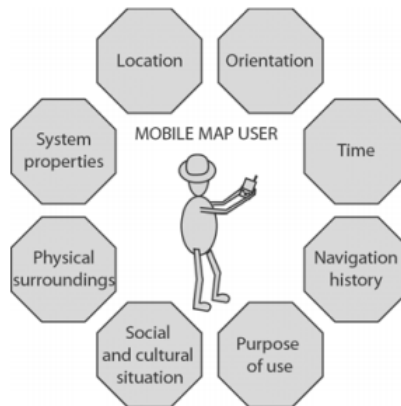


Figure 4.3: Factors that can be taken into consideration by a context-aware application (Nivala and Sarjakoski, 2005a).

The easiest way of describing these factors is by applying examples of how they can be used. An application can use the device's camera to adjust the brightness of the screen to the user's *surroundings*. If a Wi-Fi connection is available that can be associated with a certain area, such as an airport, library or shopping mall, the application can offer help that corresponds to that *location*, such as finding a gate, a book or a store. This is also an example on how *purpose of use* can be defined. Most applications must be tailored for use on a certain software platform, which means *system properties* like what operating system the application currently runs in, needs to be taken into consideration. The GPS in a mobile device can contribute with the *location* and *orientation* of the user. Along with the current *time* and *navigation history*, the application can get a good idea of why it is being used. Other aspects that might be taken into consideration are the *social and cultural situation* of the user, even though this might be hard for the application to automatically find out.

4.5 Design and development of a mobile GIS application

Designing maps that have a large amount of usability requires an adaptation of usability heuristics. The importance of involving a user-centered approach to designing and developing applications is also described in *3 Interaction design*. This approach should also be used when creating GIS applications. Research has been done (Nivala et al., 2005b) on how to effectively design and develop GIS applications from a user-centered approach focusing on the usability of the application.

The basics of the interaction design process mentioned in *3.2 The process of interaction design* are used when developing GIS applications as well. Just like in any iterative process it is cost effective to bring in the usability aspect at an early stage. Identifying the users and their needs when it comes to GIS applications is a difficult task because of the broad range of usage for these kinds of applications. The developers need to be aware of the context of use, how the application will be used in a day to day situation, for their application. This means conducting field tests on potential users is needed to evaluate the mobile context of use, as well as having market studies conducted to bring in more information about users (Nivala et al., 2005b).

There are essentially two other aspects that need to be considered when developing a map application, the *cartography* and *GUI*. The evaluation of these aspects should be done separately. Evaluating the cartography is complicated. Care should be taken when asking users for their thoughts on cartography since it is a subjective matter. It is better to bring in cartographic experts to evaluate the cartographic design because of their knowledge of the possibilities, restrictions and spatial cognition of users (Nivala et al., 2005b). The evaluation of the GUI should be based on standard GUI usability heuristics (Nivala et al., 2005b). Usability principles such as Norman's and Shneiderman's are described in *3.4 Principles*. However, there is research done on usability for map applications that is based on Nielsen's usability heuristics (Nielsen, 1994). The following guidelines are inspired by these heuristics:

- simple and natural dialogue
- speak the user's language
- minimize the user's memory load
- consistency
- feedback
- clearly marked exits
- shortcuts
- prevent errors
- good error messages
- help and documentation.

The GUI should be kept as simple as possible and use natural terminology that the users can understand. This will make the interface more intuitive for the users which will make them more receptive towards the application. It should also be kept in mind that the interface can be adapted to different user groups, such as experts being able to use shortcuts and novices receiving help when using functionality. It also needs to be adapted to different contexts of use. There should not have to be parameters and such to establish in order to be able to use the map, it should rather be “ready to be used” from start up. The buttons and icons should be logically and intuitively formed, in addition to being consistent throughout all the states of the application. Having an application totally error-free is ideal, but often an impossible goal to reach. This means that informative error messages need to be used, along with help and documentation to find out more information about these errors (*Nivala et al., 2005b*).

Other research identify a somewhat different approach to what the fundamental issues are that need to be taken into consideration (*HaptiMap, 2012*). The three fundamental issues that in this case are emphasized are *users, context* and *technology*. It is clearly obvious that this terminology has striking resemblance to what has been presented earlier in this section. The slightly different topics of the issues can easily be fitted into the previous approach. The technology issue that was previously presented did not include problems regarding hardware, but merely problems of how to construct the software to present a well-formed user interface. In this subsequent approach similar software issues are acknowledged. Additionally a number of self-explanatory hardware technological issues are also brought up (*HaptiMap, 2012*):

- accurate positioning and directions
- sound and speech interaction
- battery preservation.

5 Technical overview

In this chapter a technical description of map application technologies is presented. Emphasis is put on application technologies that have been encountered and examined throughout the development of *iSurvey*. Not all of them are used in the development of *iSurvey* but were all considered at some point in the process. Technical solutions for developing *iSurvey* are presented in chapter 8 *Technical solutions*.

The different technologies are divided into smaller groups depending on what part of an application they concern. These groups make out the disposition of this chapter. Under each group there are a number of different technologies described. The same technology can, to a certain part, be mentioned under more than one group, for a greater understanding of how an application is built up. Technologies are described in enough technical detail to explain their functional role in a map application. The purpose of this chapter is to underpin the choices made in the development process. The group topics are the following:

- mobile hardware
- application type
- software framework
- web GIS standards
- databases and security
- programming and markup languages.

5.1 Mobile hardware

An application can be developed to run on a desktop or laptop computer, on a mobile device or both. While there are a variety of available map applications for computer use, this thesis only handles development of applications for mobile devices. Today the most widely used mobile devices are smartphones and tablets. There are other types of *pocket computers* that also have internet connection and GPS. These forms of mobile devices will not be discussed in this thesis.

5.1.1 Smartphones

A smartphone is a mobile phone, almost always with an internet connection and GPS integrated in it. Compared to ordinary mobile phones, called *feature phones*, they usually have larger screens, more powerful processors and run on a more advanced platform much like a computer's operating system. Along with the larger screen, it typically uses a touch interface. All these features make the smartphone able to handle more complex operations and view internet-based information in a more interactive manner. The operating systems simplify the process of application development for the devices by offering a standard interface for functionality.

5.1.2 Tablets

A tablet is a smaller mobile computer that usually uses a touch interface and integrated virtual components, such as a keyboard. Tablets have internet connection possibilities, GPS and may or may not be controlled by a stylus or touch pen. The idea of the tablet in comparison to a laptop computer is to increase the portability while still maintaining a great amount of power. Just like for smartphones tablets run complex operating systems, compared to other mobile devices. Hence, application development is made simpler.

5.2 Application type

Applications for mobile devices can generally be divided into three groups. The first group can be defined as original functionality applications that come with the delivery of the operating system. These are not of interest in this thesis, since an operating system is not being developed. The second group is applications available for downloading and installation, so called *native applications*. The third and final group of applications is the so called *web applications* that are run in a web browser and are connected to a remote server (*Mobithinking, 2010*).

5.2.1 Native applications

Native applications are applications that are downloaded to the device and installed to pose a stand-alone application. Note that they still use the device's resources, such as the internet connection, to perform tasks. The native app is explicitly designed to run on a certain operating system. It benefits on an operating system's graphical user interface. This means for example that it uses interface features like lists, buttons and pop ups from the operating system. This makes the application's interface look more integrated with the rest of the device's features and functions.

5.2.2 Web applications

Web apps, or browser applications, are run directly in the device's web browser. The code that runs the application is downloaded every time the application is started and therefore makes it entirely dependent on having an internet connection. The purpose of a web app is to have the *look and feel* of an ordinary native application. These applications can run on any device with a browser that is compatible with the type of code the app generates. This broad compatibility results in that web apps cannot use some of the device's assets, such as the camera, accelerometer or local storage (*Bergström, 2012*). This is because these assets differ in implementation on different types of devices. Some of the device's assets are however included in the web browser implementation and can therefore be used by a web application. Examples of this are the internet connection and the GPS.

WebKit For a web browser to render layout and perform other web based operations, a browser engine is needed. *WebKit* is an open source example of such a web browser

engine (*WebKit, 2012a*), using a *BSD license*. The technique is suitable for, but not limited to, handling mobile web browsers. It is the underlying technique for browsers such as the *Android browser*, *Google Chrome* and *Safari (WebKit, 2012b)*. Examples of browser operations that need to be handled by the browser engine are rendering a website, keeping track of history and back and forth buttons etc. The technique is also used in other web based applications, primarily on iOS (*Apple Developer, 2012*).

5.3 Software frameworks

Depending on which hardware and application types are chosen for development, there are several software frameworks available. Under this topic several of these frameworks are presented along with a description on how they may be used. There are software frameworks available for both native and web application development which means choosing a certain framework may or may not bind the development to a certain platform.

Lie Luo, head of telecom, technology and media practice at Global Intelligence Alliance believes that (*Mobithinking, 2010*)

“the most significant event that changed the whole industry was when Apple introduced the App Store in 2008 shortly after it opened up the iOS software development kit (SDK) to third-party developers”.

After the release of *iOS SDK* anyone could start developing applications for mobile devices. Since then, new ways of developing applications have arisen. Along with the *iOS SDK*, the *Android SDK* and several web app developer kits such as *Sencha SDK Tools* and *jQuery SDK* are available for application development.

5.3.1 Software development kits

A *software development kit (SDK)* is a tool used for development of applications following a specific framework. As mentioned earlier this framework might be a certain operating system or a web browser. Common tools included in an SDK are solutions for the developing environment, simulators, libraries and debugging (*Android SDK, 2012*). An SDK library is an interface of functionality that is usually termed the *application programming interface (API)*. The development environment can be either a code editor or a graphical development interface.

5.3.2 Android development

Application development for Android operating system is done in Java programming language, with the help of Android SDK. An *integrated development environment (IDE)* is available as a plugin for Eclipse, so that Android development is made possible and convenient. Android Java though, is different from the common Java language. It does not follow the Java standards, Java IE and ME. The reason for this is that Android development is meant for applications on mobile devices with limited processing power and storage capacity (*Android SDK, 2012*).

5.3.3 iOS development

iOS is the mobile operating system developed by *Apple*, that can only be used for *Apple* products, such as the *iPhone* and the *iPad*. *iOS* applications are developed in the IDE called *Xcode*. The development code used is *Objective-C* which is a language similar to *C* and *C++*. The IDE makes development of *iOS* compatible applications possible, and there is also a possibility to simulate the applications in an *iPhone simulator*. However, it is not possible to simply transfer the application software onto the device and run it or upload it to *Appstore*. For this a developer's fee has to be paid to obtain an annual license. The fees are higher if a commercial application is being developed. There is also a review process for all applications that are going to be uploaded to the *Appstore*. This means that if the application is declined, it must be revised in order to be accepted and distributed via *Appstore* (*Apple Developer, 2012*).

5.3.4 Sencha Touch and jQuery mobile

Sencha Touch is a graphical user interface used for creating mobile web applications that look and feel like native apps. Applications are run in a web browser that has to be *WebKit* compatible. The API is a language built on *JavaScript* code. An application is built up like a web page with *HTML5* structure and *CSS3* styling (*Zwick, 2010*). Since *JavaScript* is used, functionality can be programmatically customized and *Sencha* code can be mixed with any other *JavaScript* based API. This makes it possible to construct functionality that is dynamic and can communicate with remote servers. The *Sencha Touch* API script is imported in the head of the *HTML* start page. All script code present in an application is available for users to view. *Sencha Touch* is an open source library (*Sencha, 2012*).

Much like *Sencha Touch*, *jQuery mobile* is a *JavaScript* library with a graphical user interface for mobile web application development. It, too, is run in a web browser and is built up by *HTML5* and *CSS3*. Unlike *Sencha Touch*, *jQuery* runs in all the modern versions of the most common web browsers, and not only in *WebKit* based browsers. *jQuery* can handle *Document Object Model (DOM)* standard which can be used when parsing responses in *eXtensible Mark-up Language (XML)* form. It is well integrated with dynamic remote server communications, like *Asynchronous JavaScript and XML (AJAX)* (*jQuery Mobile, 2012*).

5.3.5 OpenLayers and Leaflet

Open Source Geospatial Foundation (OSGeo) currently runs a project called *OpenLayers*. It is an open source *JavaScript* API used for map integration in web applications and web pages. Viewing and writing geospatial information is made possible by implementing for example *Web Map Service (WMS)* and *Web Feature Service (WFS)* functionality. These services are described in greater detail under *5.4 Web GIS technologies*. Since the API is built on *JavaScript* it is possible to implement both customized functionality and customizations of already existing code. The technology can be used free of charge and is used by both professionals and hobbyists.

There is a wide span of available existing implemented functionality, all of it documented on the official OpenLayers web site (*OpenLayers, 2012b*). The web site also has forums for discussing encountered problems by users and missing functionality. OpenLayers update their library to integrate it with other libraries and post implemented examples of this on a web page (*OpenLayers, 2012a*). Integrated functionality includes importing maps straight from, among others, *Google, Open Street Map* and *Bing Maps*. Mobile use is supported by the OpenLayers API . There are implemented examples of Sencha Touch and jQuery mobile on the examples web site, to make it easy to initiate development of web map applications (*OSGeo, 2012*).

Leaflet is an alternative JavaScript library for displaying maps from remote sources. It is built up by using HTML5 and CSS3, which makes customized styling possible. This also makes the graphical map interface smooth when panning or zooming. Compared to OpenLayers, Leaflet is not as well established yet, and is still missing some basic functionality in its library. The script file that has to be downloaded at application start up is made very slim for fast rendering of the starting page of an application using Leaflet (*Leaflet, 2012*).

5.3.6 Open source licenses

Open source libraries are meant to be open to and be used by anyone and in that way become better by users improving the code. For this to work, rules for using and sharing open source software must be defined. In this section the open source licenses are shortly described, that are used for the libraries presented in this chapter.

Sencha uses a *GNU General public license*. This license is a very free license that allows a user to use the software for any purpose, change it, share it and share the changes made. A user cannot restrict a product developed from a software under a GPL, with another stricter license (*GNU, 2012*). The newly developed product must in this case also be under a GPL license.

jQuery uses the permissive *Massachusetts Institute of Technology (MIT) license*. Permissive means minimum limitations are put on usage and redistribution of the software. This license does not limit any form of use of the software it regards, as long as the new software is also published under this same license (*Open Source Initiative, 2012a*). It may be integrated with proprietary software licenses, in which case the proprietary software part remains under the same license as it was before.

Both OpenLayers and Leaflet uses *Berkley Software Distribution (BSD) licenses*. OpenLayers uses the *FreeBSD* license which is a simplification of the BSD license. BSD licenses are permissive, like the MIT license (*Open Source Initiative, 2012a*).

5.4 Web GIS standards

To view, query and write information to and from a geospatial database there are several protocol standards that can be used. The protocols presented in this thesis are the following:

- Web Map Service (WMS)
- Tile Map Service (TMS) and Web Map Tile Service (WMTS)
- Web Feature Service (WFS)
- Proj4js script

The Proj4js script is not a protocol standard but a JavaScript library. It is, however, also presented under this topic.

5.4.1 Open Geospatial Consortium and International Organization of Standardization

Open Geospatial Consortium (OGC) is a consortium built up by companies, government organizations and universities. The purpose of the consortium is to build up world-spread standardizations of interface implementations within the geospatial field (*OGC, 2012*). The standards are then to be used in application development projects.

The *International Organization of Standardization (ISO)* is the leading organization for international standards development. Specifying standards for all sorts of products ensures the safety and quality aspects of them (*ISO, 2012*). ISO specifies these standards with the help of professionals working with the product at hand, that is in need of a standard.

5.4.2 Web Map Service

WMS is a protocol used for viewing and querying geospatial information. The protocol is usually implemented as three types of operations to be sent as requests to the WMS server. These requests are *GetCapabilities*, *GetMap* and *GetFeatureInfo* (*OGC, 2005a*).

If a list of what information the WMS server carries is needed, a *GetCapabilities* request can be sent to the WMS server. In this request no parameter values need to specify details on map layers or extent (*see Table 5.1*). The request simply inquires the general information and capabilities of the service at the given location in the request string. The response is often used to formulate the content of the *GetMap* request string. The *FORMAT* parameter, although not mandatory for this request, can specify on what mime type form the response to the request will be packaged in. The response is usually an XML file, which is a format that can be conveniently parsed (*see section 5.5.1 eXtensible Markup Language*). The response is built as a tree of object information that is simple to navigate through. Information can for example be found about

every single layer attribute, if it is queryable or opaque, etc (OGC, 2005a).

The GetMap request is used for map viewing purposes. A request is sent to the WMS server location, as a query string containing values on specific predetermined parameters. What values these parameters are given depends on what map image is requested from the user. The mandatory parameters for the different types of WMS requests are presented in *table 5.1*. When the request reaches the WMS server the server constructs an image of the specified format type containing a generated map that corresponds to the specified request parameter values. If the request is faulty or no information exists that matches the requested parameter values, an exception should be thrown and returned by the server (OGC, 2005a). An example of a GetMap WMS request:

```
http://maphost.com/wms_service_location?VERSION=1.3.0
REQUEST=GetMap&LAYERS=buildings&STYLES=iSurvey_default
&WIDTH=640&HEIGHT=320&FORMAT=image/png&SRS=EPSG:3009
&BBOX=-180.0,-90.0,180.0,90.0
```

Table 5.1: *Mandatory parameters for WMS requests.*

Request	Description	Mandatory for
VERSION	The protocol version number	GetMap, GetFeatureInfo, GetCapabilities
REQUEST	Type of request	GetMap, GetFeatureInfo, GetCapabilities
STYLES	List of styles for the map layers	GetMap
LAYERS	List of map layers	GetMap
SRS	Coordinate reference system	GetMap
BBOX	Bounding box	GetMap
WIDTH	Pixel width of map image	GetMap
HEIGHT	Pixel height of map image	GetMap
FORMAT	Map image format	GetMap
Map request part	Copy of specified parameter values from the original GetMap request	GetFeatureInfo
QUERY_LAYERS	Map layers to be queried	GetFeatureInfo
INFO_FORMAT	Mime type of return information	GetFeatureInfo
I	Pixel coordinates	GetFeatureInfo
J	Pixel coordinates	GetFeatureInfo

The last WMS request type is GetFeatureInfo and is used to query features in a map layer that is otherwise not interactive. The response to the request is generally an XML file containing information about all features, possibly from different layers, that correspond to the request parameters. Part of the request string consists of parameter values from a previously generated GetMap request (*see Table 5.1*). The rest of the mandatory parameters define where in the picture and in which map layer(s) the feature(s) of interest is situated. The response includes all available information, offered by the server, about the feature. In the application receiving the response it can then be handled and only the information of interest extracted (*OGC, 2005a*).

For all three of these request types there are several other available optional request parameters. Among these there is an EXCEPTION parameter that is important to implement so the application sending the request does not crash.

5.4.3 Tile Map Server and Web Map Tile Service

Tiles refer to the parts in a map image pyramid that represent a geographical area on more than one zoom level (See Figure 5.1). The difference between tiles and images generated by a web map service is that the tile images are part of a pre-generated pyramid on the server and can therefore be fetched much quicker (OGC, 2010). Since two requests for an image, covering the same area on the map, will always result in a response of the exact same pre-generated tiles, it is not necessary to request a tile from the server each time it is needed. Therefore caching tiles comes naturally for this type of map retrieving technique.

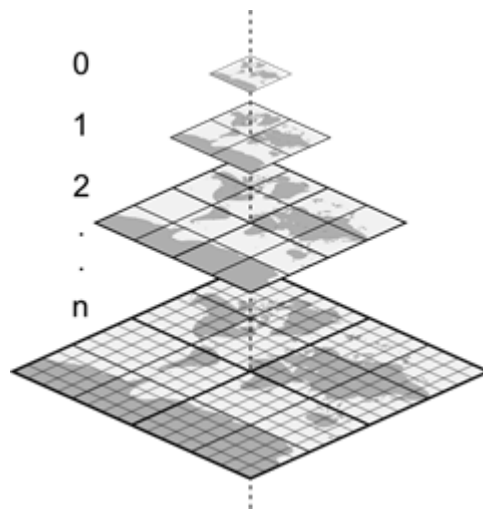


Figure 5.1: A tile pyramid. Every square tile in a pyramid has the same amount of pixels, but presents a view of a different extent of the map (WebGL Earth, 2012).

Tile Map Server (TMS), is a protocol that handles tiled map information by defining a set of available scales. These scales can carelessly be compared with a zoom level. For each scale there is a defined tile matrix carrying map information in an organized manner, in respect to positioning on the map. The information stored in the matrix is tiles, presented as map images. The idea of TMS is for the user to only be able to request map information that is pre-generated on the server. This enables the possibility of caching information, which means saving it in the web browser storage (OSGeo, 2012b), (Maso et al., 2010).

The *Web Map Tile Service (WMTS)* is a standard that handles WMS-like features such as requesting a map view, map metadata and feature info, and caches the response to minimize the weight put on the server. Instead of the server generating a map image to correspond to the request, pre-generated tiles are sent. This also means that cached information can later be used in offline mode. The request types for WMTS are similar to the WMS request types GetMap, GetCapabilities and GetFeatureInfo, except for

viewing the map. The request type for that is *GetTile* (OGC, 2010; Maso et al., 2010).

TileCache TileCache is an implementation of the TMS technology (TileCache, 2012). It is built up by server and client software that handles requests and responses. TileCache uses different types of requests. For example WMS like request strings or TMS tile requests are used. OpenLayers has functionality that acts as TileCache client software. The server side TileCache software usually resembles an ordinary file systems with the tiles sorted into different folders at different folder depths to simulate the different scales available in a tile pyramid. This software helps generating tiles from for example an existing WMS source (Nordström, 2012).

5.4.4 Web Feature Service

Previously described technologies for handling geospatial information only include fetching the information from databases as *read-only* operations. Getting and querying map information can also be done by using a *Web Feature Service (WFS)*. For writing geospatial information to a database, a different standard is used, *Transactional Web Feature Service (WFS-T)*. This standard implements feature operations such as create, delete, update and lock information in a database. With WFS-T operations, functions in an application can manipulate information in an already existing database (OGC, 2005b).

Another substantial difference between WFS and WMS, apart from WFS-T being able to write to a database, is that a WFS response to a map request is sent as geographical objects. These objects are as a default packaged as the file type *Geography Markup Language (GML)*, which is a dialect of XML that handles geographical information. WFS responses may also be sent as any other file type that can represent geographical objects. The fact that objects are recognized on the client side of an application means that the features seen in the map are properly interactive and not just part of an image as in a WMS. In a WFS implementation operations for getting and querying information must be implemented as a minimum, along with an operation for what XML schema (or file type) the response is sent in. These operations are *GetCapabilities*, *GetFeature* and *GetFeatureType*. Implementing the WFS-T operations *Transaction* and *LockFeature* is optional (OGC, 2005b). The ability to write information onto a database is due to the underlying fact that responses are built up by geographical objects instead of map images.

5.4.5 Projection script

For managing cartographical projections in a web map application, projection scripts are used. *Proj.4* (OSGeo, 2012c) is a library handling cartographical projections, and *Proj4js* (OSGeo, 2012d) is a JavaScript implementation of the library. These scripts enable the possibility to transform features' geospatial data from one projection to another. There are also scripts defining every single projection, such as *SWEREF 99 15 00*. A projection coordinate system is defined in programming scripts by a certain code, for example *EPSG:3009*. Such scripts, defining coordinate systems, can

be downloaded and integrated with the Proj4js script to make the application features handle the same projection as the fetched geospatial information.

5.5 Databases and security

Information can be stored in many ways, but when a larger amount of information is to be stored, databases are commonly the best solution. Databases provide organized storing of huge amounts of data, and links between different parts of data to form information with a semantic meaning. Database technology is shortly described under this topic.

When handling important information stored in databases, security becomes an issue. The owner of a database wants to provide easy access from remote sources that are meant to use the information while denying other remote sources access. In the second part of this topic, network security is further described.

5.5.1 Relational databases

The most common way of storing data in a database is through using relations in a *relational database*. Data is stored as relations (tables) that have tuples (rows) and attributes (columns). Data in one table can be linked to data in another table through primary and foreign keys. The databases are managed by software called the *Database Management System (DBMS)*, and more precisely the *Relation Database Management System (RDBMS)* for relational databases. There are several different implementations available of this kind of software, both proprietary and open source. Also, they both often use *Structured Query Language (SQL)* as the language with which to communicate with the actual database. Databases can use indexing to make retrieving data less cumbersome for the DBMS (*Coronel et al., 2010*).

5.5.2 Object-relational databases

While relational databases store information in tables, *object-oriented databases* store information in a software-like structure, with *objects*, *classes* and *inheritance*. It is more efficient to store complex information with this method than in text fields. There is however a middle path of implementing a database that holds substantial amounts of both text based information and more complex information, *object-relational databases* (*Coronel et al., 2010*). These databases implement complex object functionality into a relational-like database structure. Information is handled through SQL commands, which can also handle object-oriented features like objects, classes, methods and inheritance. In this type of implementation of a database the advantages are used of the well-structured and easy accessed relational database at the same time as object oriented functionality.

The complex information represented as objects in the database types mentioned above, can be built up in several ways. *Abstract data types (ADT)* may be used for this purpose. An ADT is not a certain data type. It is a definition of its functionality that gives

it a semantic meaning. Information about a complex object might best be presented as for example a *tree*, *list*, *stack* or *queue*. These data types are denoted abstract, since they are built up by primitive data types structured in a certain way, such as *integers*, *characters* and *booleans*. The primitive data type values hence get a semantic meaning when organized in this manner (*Burleson Consulting, 2012*).

Spatial databases Relational databases are effective at storing data that can easily be expressed as single text fields, however, they are not optimal for storing geographical objects. Map feature attributes are usually text fields bound to a certain object, but the geometry of the feature is not easily represented in a text cell of a table. Instead, *spatial databases* reserve an attribute cell of each tuple to point to the feature's geometry that is actually stored somewhere else, in a more complicated form (*Brimicombe and Li, 2009*). They are usually implemented as object-relational databases. Spatial databases are handled by SQL commands and can be indexed. In addition to storing location based data, they can also perform location based operations on the data, such as calculating distances between objects. There are defined standards of this type of additional handling of geometrical functionality to a database system, examples being *Simple feature access* and *well known text (WKT)*.

5.5.3 Database transaction

One major advantage of network database connections is that the same information can be fetched from several sources, a property that can also result in database security issues. Another issue is when a database operation is interrupted, by for example a power outage. *Transaction* technique is used to prevent this. The two main purposes of database transactions are (*Coronel et al., 2010*):

1. Prevent errors from occurring in the database information due to failure of completing all in a series of coherent operations on the database.
2. Prevent errors from occurring in the database information due to simultaneous write access to the database from applications run on different devices or applications.

A well-functioning transaction feature in a database fulfills these purposes by having the properties *atomicity*, *consistency*, *isolation* and *durability (ACID)*. *Atomicity* refers to the concept that in a series of coherent database operations, all or none of them should be successfully completed. Before these operations are started, a transaction is opened and the operations are temporarily performed. The result can then either be reset or properly performed, through the commands *ROLLBACK* or *COMMIT*. In any case, be it reset or performed, the database will still remain in a valid state and will therefore always be *consistent*. The property *isolation* refers to that concurrent executions of operations will, even though simultaneous, be virtually performed as serialized. This is to make sure they will not intervene with each other. After an operation is committed and the database has been changed, there is no way of undoing the operation, which means that the *durability* property is true (*Gollmann, 2011*).

5.5.4 Firewalls

In computer networking, security issues are solved using firewalls. Issues occur due to the existence of several, for the user, available networks. Some of the networks, to which a device is connected, are not secure, while other networks contain valuable information with respect to integrity or confidentiality. A firewall is therefore set up as hardware and/or software to work as a filtering bridge for communications between networks. The firewall processes information in both directions, devices on the inside trying to reach a network on the outside, as well as devices on the outside trying to reach the information on the inside. For a firewall to work, all traffic must pass through it. It should also handle different types of networks separately. An organization can for example use both a *local area network (LAN)* and *Wireless LAN (WLAN)* (Gollmann, 2011).

5.5.5 Perimeter networking

A firewall is effective at blocking unwanted sources to gain access to certain information. But what if a service from the outside needs access to both internal and external connections, and have information be sent both ways. For security reasons all transfers need to pass through a firewall. For this a *demilitarized zone (DMZ)* can be created as a perimeter network. The DMZ lies inside a firewall that allows access from the remote sources, but outside a stricter firewall (see Figure 5.2). The requested information can be pushed from database software lying inside the inner firewall, to a database within the DMZ, and can then be accessed by a service. The service can now be accessed from both the outside, through the outer firewall and still handle information from the internal network (Gollmann, 2011).

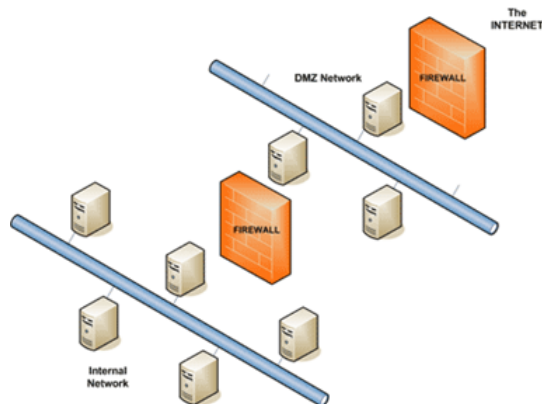


Figure 5.2: The DMZ network is set up as a halfway between an internal network and remote sources (Techrepublic, 2012).

5.5.6 Database products

There are a number of *DBMS* products available. In this section two database implementations are presented. One of the databases is an example of an object-relational database, while the other is an example of an ordinary relational database.

Oracle Spatial An example of a spatial database is *Oracle Spatial*. It provides a geographical data storage extension to the ordinary Oracle database system, but is otherwise handled in the same way as the regular *DBMS*. Data can be retrieved, updated or queried (*Brimicombe et al., 2009*). Oracle Spatial is also compatible with setting up services to communicate with the database over a network, such as the technologies described in *5.4 Web GIS Technologies*. Oracle Spatial supports database transactions (*see section 5.5.3 Database transaction*), (*Nordström, 2012*).

Microsoft Access *Microsoft Access* is a relational database based on the *Microsoft Jet Database Engine*. It uses a graphical user interface for building databases and managing information in the database. The purpose of the program is to simplify database management by allowing user to customize the database after the specific needs.

5.6 Programming and markup languages

For a web application to include a broad spectrum of functionality and remote connectivity, it cannot be programmed using only one programming language. In this review several libraries and interfaces, described in chapter 5 are based on the same language, JavaScript. For an entire application to run, however, other code languages are also needed.

5.6.1 HyperText Markup Language 5

HyperText Markup Language (HTML) is a markup language used for presenting information, generally in a web browser. The language is built up by tags used to modify how text is displayed. *HTML5* is the latest version of this language but is not yet standardized (*W3C, 2012b*). Instead it might be defined as a *living standard*, which means it is constantly renewed and revived. The purpose of *HTML5* is to evolve *HTML* code to integrate new dynamic functionality. It is necessary in mobile web application development, to get a fitted look and feel. In a web application the *HTML* code makes out the *structure* (*Freeman and Robson, 2011*).

5.6.2 JavaScript

JavaScript is a script language used to make web site applications dynamic. When a web site that runs a script is started, the script is downloaded to the client side. The script can either be imported from another file in the head of the *HTML* file or fully written there. The code is interpreted by a web browser that does not need any plugins,

generates dynamic functionality and returns it in HTML form. APIs are usually interfaces or libraries and are often implemented in JavaScript code. This is possible because JavaScript is a very open language while still providing so called first-class functions. When a function's arguments or parameters can be other defined functions, the language is said to have *first-class* functions. This makes the script language JavaScript much like an ordinary programming language. In a web application the JavaScript code makes out the *behavior* (Freeman and Robson, 2011).

5.6.3 Cascading Style Sheets

Cascading Style Sheets3 (CSS3) is the latest version of CSS, and is used to design web pages. Developers have free control of how thick borders should be, what fonts should be used, use of backgrounds, etc. It can be used together with any XML-based language. CSS3 and the markup languages are separated, which makes it easier to make changes to either design or content of a website, without having to change the other one. It is also easier to share CSS3 with others and other websites (W3C, 2012a). Open source graphical interface APIs often have predefined CSS integrated with their JavaScript library. In a web application the CSS3 makes out the *presentation* (Freeman and Robson, 2011).

5.6.4 eXtensible Markup Language

eXtensible Markup Language (XML) is a markup language used to carry semantic data in a structured form. The syntax for this language is much like the syntax for HTML, with start tags, end tags and the tagged information within. XML tags may be customized in any way, unlike HTML that can only use standard tags. This might give the idea that XML is an uncontrolled language, which is not the case, due to the requirement that XML files must be well-formed. There are a number of specified rules that must be followed (Harold, 2004):

- No nested tags are allowed.
- Document must contain only one single root.
- Tags are case sensitive.
- Values of attributes must be written in quotes.

XML is a metalanguage that allows for dialects of it to be developed to fulfill a certain purpose. The rules of a certain dialect (or schema) are usually described in a *document type definition (DTD)* or an *XML schema definition (XSD)*. If an application uses several XML files that might contain same named tags that have different meanings, *namespaces* are used. For example a *heading* might refer to a text topic or the course of a ship. If both these semantic meanings have to be represented at the same time, namespaces are used to differentiate them by adding a name prefix. The element heading is divided into text:heading and ship:heading, to separate the semantics of the two elements (W3Schools, 2012). If namespaces are used in an XML file, the namespace definition is usually linked in the beginning of the document.

5.6.5 ASP.NET technology

Active Server Pages .NET (ASP.NET) is a framework used in development of dynamic web services and applications (*Microsoft, 2012*). The dynamic execution of an ASP.NET service can be initialized within a JavaScript. A new web page is opened containing the *inline code* in HTML form. This is the visible code for the user. Behind this code is the server side script that actually performs the dynamic function, the *code behind*. The fact that the script remains server side (hidden from the user) is a substantial difference between this technology and JavaScript. Therefore, ASP.NET scripts are more suitable when handling sensitive tasks, such as database or user login operations. The code behind script is written in either *C# (“c sharp”)* or *BASIC*. C# is an object-oriented programming language that handles most functionality expected by a fully evolved language (*Evjen et al., 2010*).

5.6.6 Asynchronous JavaScript and XML

Asynchronous JavaScript and XML (AJAX) is a combination of several techniques that allows seamless interaction between the user and an application by adding an AJAX engine as a middleman between the application and servers. This interaction is asynchronous, meaning the user can use the application independently of the communication with the server (*Garrett, 2005*). In comparison to for example ordinary JavaScript, the AJAX script execution is sent to a location outside the application to be performed without further disrupting the application code execution.

5.6.7 Well Known Text

Simple Features is a standardized way of representing geometrical objects. Part of the Simple Features representations is the *Well Known Text (WKT)* standardized text markup language. It is used to present geometry in a standardized form that is easy to create and parse inside application code. WKT can present all features defined in the Simple Features specification. Representation of single points, lines and polygon looks like this:

```
POINT (X1 Y1)
LINESTRING (X1 Y1, X2 Y2, X3 Y3)
POLYGON ((X1 Y1, X2 Y2, X3 Y3, X1 Y1))
```

If several object of the same type are to be written on WKT form, this is possible by using *MultiPoint*, *MultilineString* and *Multipolygon* (*OGC, 2010*). This type of geometry representation can easily be used to create a relational spatial database. This approach is good for storing data but, not as suitable for performing database side spatial operations.

6 Needs analysis - iSurvey

A presumed need exists for a map application that is able to run on mobile devices out in the field. The basic outline of this application should be a digital map that would substitute the printed paper maps in the field. It should show map layers, carry information about map layers and have some kind of drawing feature.

The head of the *Map and surveying unit* at *Örebro city planning department*, who initiated this project, believes that a need for an application like the one mentioned above exists. Thus, a needs analysis was conducted and is described in this chapter. This analysis is supposed to determine if there is a need of such an application as described above.

6.1 Methods of analyzing

Information was gathered to identify the general needs of an application but also, in greater detail, identify desired specific functions. Gathering of information was done through meeting with employees that are possible future users.

Meetings were held with employees from several units within *Örebro municipality* who had registered their potential interest for an application. In this way expected users were identified. The units are *Lantmäterienheten (Administering land survey unit)*, *Miljökontoret (Environmental unit)*, *Kart- och mäthenheten (Map and surveying unit)* and *Bygglövsenheten (Planning permissions unit)*. The purpose of these meetings was to identify the specific needs of that unit through a user-centered approach. First of all it was to determine if there is a need at all for an application. Secondly more specific functionality was discussed.

Meetings were held between the developers (authors of this thesis) and representatives from each unit. A basic outline of the application was presented to the representatives. They then described how they conduct their field work and considered if an application could help. They were then allowed to put forth their own specific needs for doing inventory work in the field on a mobile device. Brainstorming methods were used. In this first stage all ideas were taken into consideration, as long as they somewhat coincided with the basic purpose of the application. The meetings were held as very open interviews.

All the gathered needs were then compiled and compared to one another and ranked according to their importance and to which realistic degree they could be implemented within the time span given. After this a preliminary needs analysis was made. The construction of the needs analysis is an iterative process. This means the needs analysis was regularly presented to the expected users so they could give feedback on it. This was mainly to ascertain whether they would be able to find use for an application of this structure, but also to check that no misunderstandings occurred during the first meeting. Some desired needs were instantly dismissed, due to the fact that they were too far from the scope of the application.

6.2 Identified needs

All identified needs from the interviews are listed below. Note that some needs that were brought up during the interviews were dismissed directly due to the fact that it was obvious that they did not coincide with the functionality of this application. Some of the needs that were identified during the interviews are needs that are equally useful and important for several or all units. Some needs, however, are more specific to only one of the units. General application needs:

- All units agree there is a need for a dynamic field map.
- All units agree that this application should be able to replace the printed maps in their field work.

Functional needs for the application:

- There is a need for making notes and drawing vector features in the map.
- There is a need for importing and viewing certain layers in the field.
- There is a need for viewing certain attributes that are bound to an imported layer in the field.
- There is a need to be able to turn layers on and off.
- The planning permissions unit wanted to be able to take and handle geographically bound photos.
- There is a need to be able to change the shape of drawn vector features.
- The surveyors wanted the application to handle transferring drawn vector features to the computer, both as a file of some sort and into the general GIS-system (GeoBas).
- There is a need for knowing the current GPS accuracy.
- The surveyors wanted to be able to track a distance while walking.
- The Environmental unit and the surveyors wanted to be able to see coordinates, lengths and areas of drawn vector features in the map.
- The Environmental unit also wanted to be able to use different colors when drawing vector features.
- The surveyors wanted the possibility to search for property units and addresses, and to view them on the map.
- Since the internet connection has varying coverage in the field, there is a strong need for a cache-function. Employees need to be able to save (cache) a wanted extent of the map on the device before going out to do field work.
- The planning permissions unit wanted geographically bound photos to have a date stamp.

- The planning permissions unit stressed that the application needed to be interactive and easy to use, even for employees who might struggle with technology.
- The Environmental unit wanted there to be different profiles to choose from, for the different units, to meet their needs.

6.3 Dismissed needs

During the interviews several needs was brought up that could not fit into this version of the application. Those needs could probably be implemented to run in a mobile tablet application, but would be better as a completely different application. Some of these ideas were too unrealistic and far from this application's functionality that they will not at all be brought up in this thesis. The ones among the dismissed ideas that were somewhat coinciding, however, were the following:

- Being able to connect the tablet to the surveying equipment and transfer information.
- Performing in-field Helmert transformations.
- Automatically stamp an in-field taken photo with the property id corresponding to where it was taken.
- The Environmental unit wanted a digitized version of their in-field protocol form.

7 Requirement specification - iSurvey

The requirement specification chapter specifies functional requirements and performance requirements for the application that is going to be developed. A minimum set of functions and requirements are going to be listed, as well as a list of desired functions that deviate slightly from the core purpose of the application which should be implemented if time is given. The development of the application will be based on the requirements listed in this chapter. Requirements will be listed with a priority.

7.1 Required functionality of the application

Based on the needs analysis that has been conducted, a set of required and desired features was put together to match these needs and fulfill them. These features are listed below as *software requirement (SR)* and a number, followed by an explanation for each of them.

- First priority
 - SR1.1 Retrieving and displaying map layers and information from a geo-database.
 - SR1.2 Draw features.
 - * SR1.2.1 Attach note attribute to drawn feature.
 - SR1.3 Query objects in the map.
 - SR1.4 Handle the geodetic reference system SWEREF 99 15 00.
- Second priority
 - SR2.1 Allow for photos to be taken when out in the field.
 - * SR2.1.1 Taken photos should have a date stamp.
 - * SR2.1.2 Photos should be geotagged.
 - SR2.2 Show the current accuracy of the GPS receiver.
 - SR2.3 Tracking and logging current position (as a point).
 - * SR2.3.1 The GPS accuracy at the moment of tracking and logging should be saved.
 - SR2.4 A caching feature.
 - SR2.5 A search feature.
 - SR2.6 Exporting drawn vector layers and notes to a computer.
 - SR2.7 Include different profiles for the different units at the department.
- Third priority
 - SR3.1 A selection of colors for the drawn vector layer.
 - SR3.2 Tracking and logging line segments.

- * SR 3.2.1 The GPS accuracy at the moment of tracking and logging should be saved.
- SR3.3 Show coordinates of drawn features.
- SR3.4 Be able to calculate the length of a line.
- SR3.5 Be able to calculate area of a polygon.
- SR3.6 Be able to change the shape of a line and polygon after they have been drawn.
- SR3.7 Export drawn vector layers as a construction layer in GeoBas

7.1.1 First priority - Minimum requirements

The minimum requirements of the application should be implemented as the core functionality. The core functionality is the functionality that makes this application suitable as an in-field inventory map application. These minimum requirements are defined below and make out the first priority of implementation. These are features that have the highest priority and must be implemented.

SR1.1 Retrieving and displaying map layers and information The application should be built up two base map layers, a base map and one with orthophoto. The base layer should be built up by general layers for orientation on the map, for example roads, railroads, land use areas, as well as text labels connected to visible features. Property borders should be visible at some zoom levels. On top of the base layers the user should be able to add more layers from the department's geodatabase. Since retrieving layers can be very bandwidth-dependent, which makes panning and zooming on the map very heavy operations that consume a substantial amount of time, it should be possible to turn these layers on and off. The layers should be for viewing purposes only, which means no editing can be performed on them and written to the geodatabase.

SR1.2 Draw features The user should also be able to make their own notes in the map. For this a feature of editable vector layers should be added. Points, lines and polygons should be able to be drawn in the map with a possibility of a note attribute attached to every feature (SRS1.2.1). With this attribute, the user should be able to write free text linked to a certain object. The user should be free to use the text attribute for the desired purpose, the note should therefore not be controlled by adding it to as an attribute with a certain label.

SR1.3 Query objects in the map Almost all layers should be interactive in the sense that they should be queryable. This means the layers should include a selection of their attributes, which should be retrievable in the application.

SR1.4 Handle the geodetic reference system SWEREF 99 15 00 Since the department uses SWEREF 99 15 00 as the standard reference system, the application must be able to handle it. There is no need for the application to be able to handle any other reference system.

7.1.2 Second priority

These features have the second highest priority and should be implemented as build-on features/functions to the minimum requirements.

SR2.1 Allow for photos to be taken out in the field The application should handle starting the camera and allowing for photos to be taken out in the field for documentation purposes.

SR2.1.1 Taken photos should have a date stamp Taken photos should have date stamps added to them since regular photos taken with a mobile device do not display these.

SR2.1.2 Photos should be geotagged Photos should be geotagged so tracking the position of where the photograph was taken is made simple.

SR2.2 Show the current accuracy of the GPS receiver When using the application a field displaying the accuracy of the mobile device's GPS receiver should at all times be shown. This is to assist in making educated decisions when using other features of the application that rely on the GPS.

SR2.3 Tracking and logging current position (as a point) When out in the field the user might come across simple objects that should be mapped. An easier and faster way than trying to draw the point on the map is to, via the GPS, track the user's position and log it on the map in the form of a point vector object.

SR2.3.1 The GPS accuracy at the moment of tracking and logging should be saved When using the feature mentioned above, it will not suffice to only include the position of the user. The accuracy of the GPS at the moment of logging the position should also be noted, in order for the user to later make informed decisions about an object's real world position when back in the office.

SR2.4 Caching feature When moving in the field there is a risk that the coverage will be bad and that the device will not have internet connection. To deal with this a caching feature should be added. This will allow for the user to, while in the office, cache an extent of the map with layers in the application which the user later can load in the field without having access to the internet.

SR2.5 Search feature When out in the field (or before in the office when caching is needed) an easy and fast way to find the area of interest is to search for an address or property id. Such a search feature should be implemented. The search feature should allow the user to select from searching for an address or a property id, and when the search is completed, automatically bring the user to that address or property on the map.

SR2.6 Exporting drawn vector layers and notes to a computer After field work is finished and the user has drawn vector objects and written notes, a feature to transfer those objects and notes to a computer should exist. The vector objects should, when transferred, consist of positions (x and y coordinates) and the notes will be text. They should be exported into a suitable file type containing the information gathered in the field.

SR2.7 Include different profiles for the different units at the department When logging in, the user should specify a profile corresponding to which department the user belongs to. Certain profiles contain certain layers, layers that are more interesting and relevant to that profile. This is to increase efficiency out in the field.

7.1.3 Third priority

These are the least prioritized features and should be implemented if time allows.

SR3.1 A selection of colors for the drawn vector layer A way to clarify different objects in the field is to use different colors when drawing them so they are easily distinguishable from one another. The user should have a pallet of a few different colors to choose from to further simplify and assist field work.

SR3.2 Tracking and logging line segments Much like the function SR2.3, but instead of taking a “snapshot” of the user’s position, the GPS records a longer session. This allows tracking and logging of a distance that the user travels as a line segment on the map and save its position.

SR3.2.1 The GPS accuracy at the moment of tracking and logging should be saved When using the feature mentioned above, it will not suffice to only include the position of the user. The accuracy of the GPS at the moment of the logging session the position should also be noted, in order for the user to later make informed decisions about a line object’s real world position when back in the office.

SR3.3 Show coordinates of drawn features When drawing a point, line or polygon the coordinates of the vertices should be able to be viewed.

SR3.4 Be able to calculate the length of a line When a line vector object has been drawn, a feature for calculating the length of the line should be implemented.

SR3.5 Be able to calculate the area of a polygon When a polygon vector object has been drawn, a feature for calculating the area of the polygon should be implemented.

SR3.6 Be able to change the shape of a line and polygon after they have been drawn When a line or polygon object has been drawn the user should be able to, afterwards, edit the shape of the object if minor change is desired, instead of having to draw the object from scratch.

SR3.7 Export drawn vector layers as a construction layer in GeoBas After field work is finished and the user has drawn vector objects, a feature to transfer those objects to GeoBas should exist. The vector objects should be exported to a construction layer in GeoBas, where further editing can be done.

8 Technical solutions – iSurvey

In this chapter, solutions for the development of *iSurvey* are presented. At first a general structure is presented for the application and for the communications over the web. Then the chapter focuses on reviewing the development process, and arguments are made concerning why certain technologies were chosen instead of others. The technologies used are described in greater detail in chapter 5 *Technical overview*.

8.1 Chosen technologies

iSurvey is developed as a web application which uses *Sencha Touch* for the graphical user interface. *OpenLayers* is used for loading the map and layers into the application. Features from the *OpenLayers* library are used for most of the functionality, either in whole or as a part of custom functionality. *Cascading Style Sheets (CSS)* is used for styling the visual appearance of the graphics. Two custom scripts handle searching for features and querying objects in the map. Most of the scripts are coded in *JavaScript*.

Communication between the application and the database servers within the *demilitarized zone (DMZ)*, as well as searches within the *Microsoft Access database* on the application server, is handled by a few *ASP.NET* scripts. The base maps are generated *tile pyramids*, stored as a *TileCache* server in the DMZ, while the layers are fetched using *web map service (WMS)* from the database server in the DMZ. Other minor technical solutions are made possible by using *jQuery*, *AJAX* and *XML* technology.

At *Örebro municipality* all the geospatial data uses the reference system *SWEREF 99 15 00*. *SWEREF 99 15 00* uses the *GRS 80* reference ellipsoid. *Proj.4* allows the application to handle this reference system and map projection.

8.2 Mobile hardware solution

Because of the need for portability the application is developed to be run on tablets and smartphones. The full functionality of the application can only be accessed when using a tablet (or desktop computer). Running the application on a smartphone will allow the user to access a slimmer version. This is because of the phone's smaller screen size which makes using and fitting all of the functionality on to the screen almost impossible. Tablet screen sizes are optimal for the type of work that is done using *iSurvey*.

All development and testing has been done on the Apple iPad, and it is the recommended device for running *iSurvey*, although fully functional on other tablets. It is important to keep in mind that for the application to work properly the tablet should be equipped with the ability to connect to wireless internet. Tablets come in several different sizes; however for optimal use the screen should have a diagonal dimension of 9 to 10 inches (23 – 26 cm).

8.3 General structure

The general structures of the application and of the web communications are described in this section.

8.3.1 General structure of the application

In the figure below the general structure of the application is presented (see figure 8.1). It is divided into larger blocks containing elements used for that certain part of the application's functionality. The blocks are tied together in a manner that represents how they communicate with each other. Below the different blocks are described in greater detail and what role they play in the application's functionality.

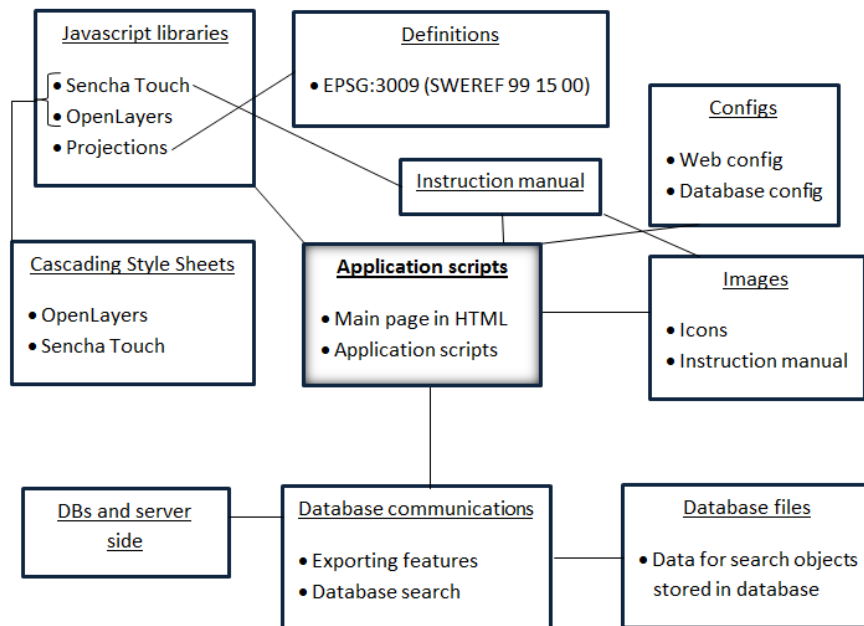


Figure 8.1: General structure of the application.

Application scripts The elements in the *application scripts* block are framework of the application. They fill the page with the graphical interface and load the maps, layers, icons and images that are used. All of the functionality is coded within these elements. The main graphical interface is built up by using Sencha Touch. The handling of map and layer information is performed through using OpenLayers' open source framework. The functions revolve around mainly using features from the OpenLayers library, but also use some features from jQuery and Proj4js.

JavaScript libraries The *JavaScript libraries* that are used are mainly Sencha Touch and OpenLayers, but also jQuery and Proj4js to a very limited extent. These have pre-written features that can be used for developing custom functionality.

Definitions The *Definitions* block contains a file that is used together with Proj4js to define coordinates in SWEREF 99 15 00.

Cascading Style Sheets Sencha Touch and OpenLayers use corresponding CSS to style the elements that are used. The elements here refer to button icons, map styles, popup menus, borders etc. In the CSS files the user has the option to modify the style of these elements. The imported CSS code that is linked to the open source libraries, is stored in entire separate files while manually formulated CSS code is included directly in the scripts.

Images The *images* block contains the icons and images that the application and instruction manual use. The icons are used for the buttons of the application and the images are used to further convey information in the instruction manual.

Instruction Manual The instruction manual is built up by a Sencha Touch interface and most of the information displayed in it is coded in HTML and integrated in to the application scripts. The instruction manual is presented in *Appendix A*.

Configs The application uses two configuration files (*configs*) for storing general application and web-based parameters, and search paths that are used by parts of the application. One of the configs concerns web site configurations while the other concerns database configurations.

Database communications The elements in the database communications block handle both communication between the application server and database server, and also application database searches. A proxy file enables communication between the application server and the WMS server, which in turn gets information pushed from the geospatial database server. There are two ASP.NET files along with their associated *code behind* files. The code behind files are coded in C# and handle:

- Exporting drawn features to the database server.
- Searching the Microsoft Access database on the application server to find the coordinates of objects.

Database files The database files are used when the search function is called. They are a Microsoft Access database that lies directly on the application server and include addresses, properties, property border points and reference point names, along with respective coordinates for all of them.

Databases and server side The elements in the *databases and server side* block represent the databases and servers of the application that contain spatial features. Two the databases are situated within the DMZ, these are the search database and the TileCache database. The TileCache database serves the purpose of acting both as a server and as a database. The third database is within the inner firewall, and contains all the geographical information. It pushes its information out to elements within the DMZ for it to be made accessible (*see figure 8.2*). The elements that receive pushed information from the inner geospatial database are the TileCache database and the WMS server.

8.3.2 General structure of web communication

There are essentially four parts that need to communicate with each other, over the internet, to make *iSurvey* operate properly. *Figure 8.2* presents the general structure of the communication over the internet and intranet. The four communication parts are:

- The web application running on a tablet.
- The application server in the DMZ.
- The database servers in the DMZ.
- The geospatial database on the intranet.

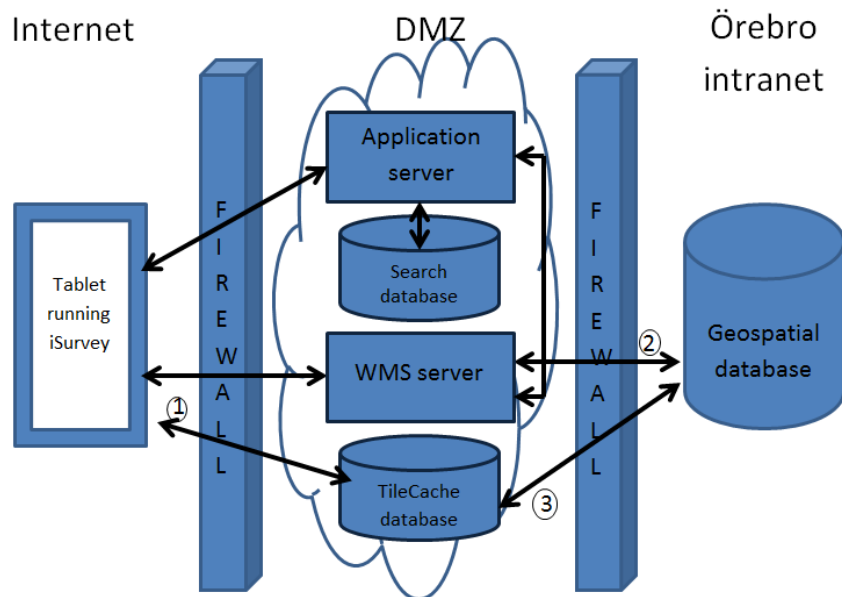


Figure 8.2: Web communication between all parts of the application. 1) The tablet browser sends requests to the WMS server (explained in detail in 8.5 Web communication solutions) or the TileCache database. 2) The information in the WMS server is automatically updated once per week. 3) The information in the TileCache database is updated manually approximately once a month.

When the web application is run on a tablet, the script files are downloaded from the application server. The application server communicates with the *search database* (used when the search function is called) which is stored in the DMZ aside the application server. The application server also handles communication with the other objects in the DMZ. The DMZ contains a TileCache database which stores the base maps that are used in the application as well as a server program that handles this database. An application runs inside the intranet that can communicate with the geospatial database. This software, linked to the geospatial database on the *intranet*, pushes information onto the WMS server and TileCache database situated in the DMZ. This makes it accessible from remote sources, such as a tablet running the application. The push that updates the WMS server is done once a week. Pushing information to update the base maps generated in the TileCache database is done manually. It's performed when it is considered required by the administrator (or on demand from an employee), approximately once a month. Pushing information must be initiated from within the intranet.

8.4 Application and software framework

iSurvey is developed as a web application. There are several reasons for this. A web application allows the application to be run on most devices that have webkit-based web browsers. There is also no need for registration on a native application market. To make a native application available to users, there is also an approval procedure which further delays the publication of the application. Making changes or updates to a web application is easy because they are made available immediately, compared to native apps that must be uploaded and then the user actively has to update it. Development of the application is done using open source software, which means that the costs for software for developing are nothing. Sencha Touch and OpenLayers are used for the greater part of the development, but several others are used for minor details. Sencha Touch is chosen because the API gives developers pre-implemented building stones for making GUI:s that have the look and feel of a native application, with extensive documentation and examples to aid developers.

OpenLayers API is used for most of the functionality of the application. OpenLayers is chosen instead of Leaflet simply because it has a few more features than Leaflet has to offer, which are crucial in the development of *iSurvey*, such as the ability to draw custom vector objects. The authors feel that the OpenLayers' documentation is somewhat lacking in detailed information, however there are many examples of feature use to aid development. One example that OpenLayers is provided is "OpenLayers with Sencha Touch" (*OpenLayers, 2012a*). This is the basic example that *iSurvey* is built on, with more functionality added.

Because the Sencha Touch and OpenLayers API:s are coded in JavaScript, it makes it easier for the developer to use both of these in conjunction with each other, rather than have to jump between different coding languages. Also, other scripting techniques used for minor functionality can be implemented or at least initialized through a JavaScript.

8.5 Web communication solutions

The nature of the application makes it rely on communication over the web. *iSurvey* utilizes a number of standard techniques for fetching geospatial data over the internet.

The base map and orthophoto are fetched from a TileCache using TMS technique. *Örebro municipality* uses a TileCache server program to generate the tile pyramids and store them in the TileCache database.

The identify function is not an OpenLayers standard function. It is mainly implemented by Sweco Position for use in similar applications, but customized for fitting *iSurvey*. The GetFeatureInfo requests are initiated by an AJAX script and sent from the application via a proxy ASP.NET script to the WMS database to fetch the content that is requested. *Figure 8.3* explains the proxy communication:

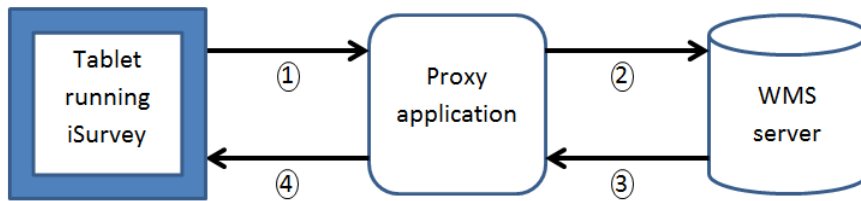


Figure 8.3: Proxies. 1) The tablet asks the proxy, via AJAX, to fetch data from the WMS database, 2) The proxy asks the WMS database to fetch the requested data, 3) the WMS database fetches and sends back the requested data as XML, 4) The proxy sends the requested data to the application where it is parsed.

For the identify function, the communication between the tablet and proxy is initiated through a jQuery AJAX script. Letting an AJAX script send the requests to the proxy allows the web application to keep running and not reload the page after each request. The request is sent back as XML. The XML is parsed and can then be displayed properly. OpenLayers is used to define WMS layers in the application. It also handles all the communication to and from the WMS database, and fills these layers with information that is fetched. Fetching layers is done by using WMS requests to send parameters to the WMS database to tell what information is to be fetched. The information is sent back to the application where OpenLayers handles the data, puts it in the appropriate layer, and displays it on the map.

There is an ASP.NET script that handles other communication. One of the scripts searches the Microsoft Access database, and is used when the search function is called. The user types in a search term, and the script searches the database for results that match the search term. These are then returned and from them a store is created and viewed in the result list. The store contains the names and coordinates of objects that were returned. Tapping on an item in the search form will then move the user with the help of an OpenLayers function to the location of that object, based on the object's coordinates.

Written notes attached to drawn features in the application are represented by a simple text field. Before implementing this function a discussion was held about what would be the best approach when saving notes. Generally, three different approaches were identified and discussed, the first being a solution with a simple text field. This implementation would mean an attribute would be saved locally in the web browser session, directly accessible when exporting. Secondly, some sort of WFS feature was discussed. In this implementation information would be written (posted) directly on features in the database. The third approach was to have predefined labels to choose from when making a note, examples would be *object type, length, position* etc.

There are several reasons why the first approach was used. WFS functionality was

not actually what was needed. Informal notes should be made on the informally drawn features. For information to be written onto the database it has to be precise and correct, and not have the informality of notes taken in the field. Another reason for not using WFS is that notes would have to be bound to a feature in the database, and just finding a feature in some sort of list when in the field would be very cumbersome. The informality of the notes also implies that they should not be controlled, if not necessary. For adding a label on a note, again, the user would have to go through some sort of list in the field, to find an appropriate one to add to the note. This would take a lot of extra time. The approach where a simple text field is added to a feature is informal and quick to write, which fulfills the needs of the employees at the different departments.

Another ASP.NET script is used when the user wishes to upload drawn features to a database. Each time the application is run, a unique session id is generated. When features are drawn, OpenLayers generates feature id:s for them, and all feature id:s will be unique during a session. The session id together with the feature id will therefore give a feature a unique id, to form a primary key in the database. The application packages the OpenLayers features into a URL request string that is read by the ASP.NET script. The script further parses this URL so that the features are represented in WKT form along with their unique session and id combination, and possible notes. The features are then inserted into a relational database to be stored. The unique session and id combination is used to make sure that features are not stored more than once in the database.

8.6 Workflow of the application script

The first script code that is run is code present straight in the start-up HTML file. This code integrates with and defines the look of imported Sencha Touch standard features to load the *graphical user interface (GUI)*. The GUI toolbar appears along with all the available buttons and status bar. After this, the imported application scripts are initiated and map servicing is started.

When initiated, OpenLayers creates a map and several layers. The map and layer objects are from the start defined with some parameters, but are initially empty. The map is defined by which projection it uses, what units are used (meters in *iSurvey's* case), what geographical bounds are used and what resolutions the different zoom levels have. The layers are defined individually since the data that is fetched is different. Examples of parameters that are defined for layers are if the layer is a base layer or not, what objects are queryable, if it should be visible on application start-up, etc.

When requests are sent and the data response is received (in the form of *png* images) the map is filled with all of the layers that data is available for and can be presented on the current screen extent. When the application is first run, the application automatically requests data about the base map, and it is that layer that the user is first presented with. Switching layers will request information about the active layers and then display them as information is received. Since the base map and orthophoto are fetched with the OpenLayers TileCache client, the application first searches in the device's browser

cache to see if the requested map extent already exists in the client.

Tapping on buttons in the application runs different methods within the scripts, as each button is tied to a particular method. The methods can be found in different scripts, but loading all of the scripts on start-up lets them act together as one script.

8.7 Design solutions

The design of *iSurvey* does to some extent follow the principles and heuristics presented in the chapters 3 *Interaction design* and 4 *Advanced interaction design for mobile GIS*. Emphasize is put on *Norman's principles*, *Shneiderman's eight golden rules* and the heuristics presented in 4.5 *Design and development of a mobile GIS application*. Examples are given of features in *iSurvey* that correspond well to a certain principle. Finally, how *iSurvey* has tackled the usability goals of interaction design is described.

The GUI of *iSurvey* implements the *visibility* principle by adding a toolbar with all available functionality accessible through it. The operations are activated by tapping a button and/or tapping straight in the map. This approach is chosen for easy access to all functions. Having a *simple* and *natural dialogue*, meaning keeping the GUI clean and simple, is somewhat set aside to achieve the easy access.

A principle that is mentioned in almost all interaction design guidelines is *feedback*. In *iSurvey*, feedback is given for every action performed: navigating the map, tapping buttons and activating functions. Messages are shown in the status field for every initiated action, along with alert and confirm messages as popups for some of the more complex functions.

The simpler smartphone version is an example of a constraint put on the application. Button symbology is constructed to follow *natural mapping* standards, with for example red buttons in the upper right corner of panels to close them. All functions are used in a very *consistent* way. They are initiated by tapping a button and then performing something on the map. When developing an application for mobile use, *affordance* never really becomes an issue, since the user of such a touch screen based device is used to tapping buttons and using general tap gestures like pinching and dragging.

Users should always *feel in control* when using *iSurvey* since the functions take a short time to accomplish their purpose and barely any completely automatic operations can be run. An exception from this is the caching feature that can take some time to complete if a larger map extent is to be cached. A confirm box warns the user that this might happen and the user is given feedback through seeing the zoom levels moving, and realize that something is happening. If at any time the user is unsure of what to do, the question mark button can be tapped to open the user manual that contains *help and documentation*.

9 The final product - iSurvey

In this chapter, the final product's functionality is described. After this description some examples of using the application are made. These examples show how the purpose of the application can be fulfilled for different municipal units.

9.1 The main graphical user interface

The application is started by browsing the application web site. When loading the website, the user is presented with the main graphical user interface (*see figure 9.1*). This viewport is mainly made up of the map, with the toolbar at the bottom of the page. Because of the application's purpose, the map has to be as large as possible but still leave room for a toolbar with buttons that are easy to tap and with icons that give a clear indication of their functionality. No button tap, with the exception of the help button and the export button, will cause you to leave the main viewport or alter the appearance of the viewport. These two buttons that make you leave the application viewport will open in a new tab. To return to the map, close the tab. The map and toolbar, and all the buttons on the toolbar, will never change size or order.

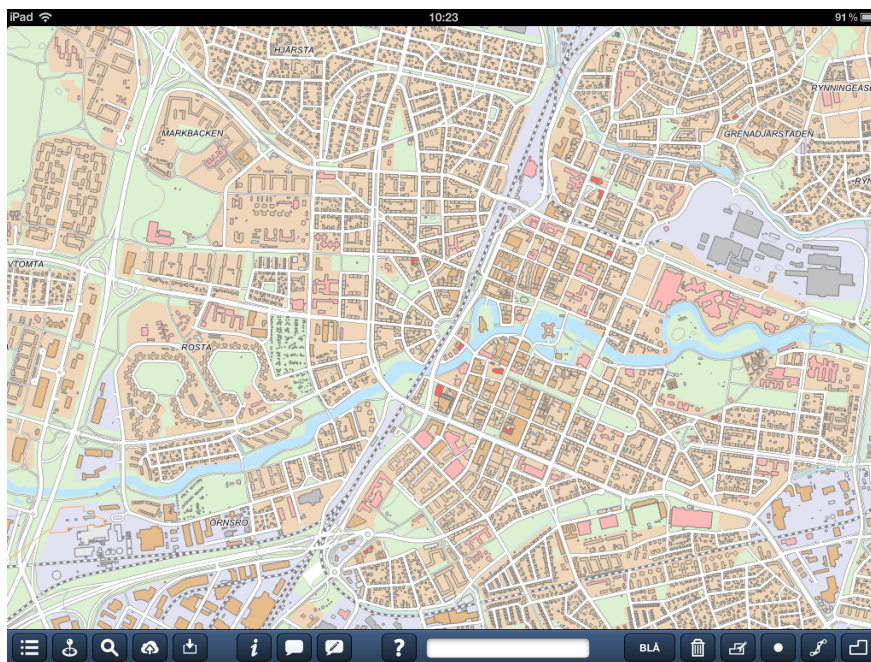


Figure 9.1: Graphical user interface of iSurvey.

9.1.1 Hardware compatibility

The application is tailored for tablet use. It has been tested on Apple's iPad iOS 5.x, but should work just as well on other tablets and operating systems. It has been run on a Samsung Galaxy Tab 10.1, but not properly tested. It should only be run in landscape mode. The application also runs on ordinary computers in the web browser, but is only compatible with Google Chrome, Safari or any other WebKit-based browser. If the application is opened on a smartphone, a simpler version of it is run, where only a selection of more basic features is available. This is handled automatically.

9.2 The map

The map displays the information that is fetched from the server databases. It displays the chosen base layer, with additional layers that have been selected by the user shown on top of it. All of the map related operations are performed on the map, such as querying objects and drawing features. Other operations mask the map when they are in use, such as selecting layers and searching for objects. The map supports touch navigation gestures. This means, on mobile devices tapping, dragging and pinching navigates the user on the map. Tapping and dragging work as one would expect a mouse to. Pinching adjusts the zoom level. Note that this is done by using the mouse's scroll wheel when in a web browser.

9.3 The toolbar

The toolbar contains buttons for activating functions and a status field (see figure 9.2). Below the buttons are shortly explained.



Figure 9.2: *Toolbar with numbered buttons.*

9.3.1 Layer switcher

Tapping the layers switcher button (1) will make the layer selector list pop up. Selecting an option sends a request to the server to fetch the information and displays it on the map. The two bold layers at the top are the base layers. Only one of these may be selected at a time. The rest of the layers are additional layers that can be displayed simultaneously and on top of the base layers. For a more thorough explanation of the map layers, see section 9.6 Layers and Table 9.1.

9.3.2 Geolocate

Tapping the button (2) uses the device's GPS and locates your position. The map will load and display your position with a red cross with a gray ring around it. The gray ring represents the accuracy of the GPS. If situated outside of Örebro municipality, one is outside the bounds of the map and this function will not work.

9.3.3 Search

A search form panel pops up when the search button (3) is tapped. In this panel the user can search for an address, a property id, a border point id or a reference point id. Tapping a search hit will center the map on that item.

9.3.4 Export

After having drawn features on the map, the user can export them to a database by tapping the export button (4). This function exports the feature's unique id, its geometry, any notes that have been written and the current session id. The user is prompted for a GeoBas id. A new tab opens in the browser confirming the export (see figure 9.3). There is also a link to a view of the contents of the database. For surveyors using this function the GeoBas id can be used for importing the features to the desktop GIS-program GeoBas.

Exporten gick bra! Stäng fliken för att återgå till kartan.

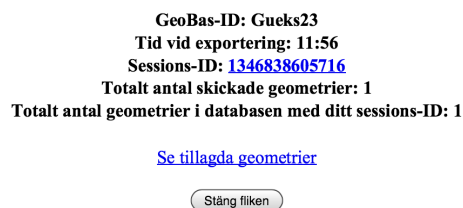


Figure 9.3: *Export confirmation page.*

9.3.5 Cache map

A caching feature is activated by tapping the cache map button (5). This caches the full extent of the current base layer along with the entire underlying tile pyramid, and allows it to be viewed offline. The cache function is a heavy operation and can take a long time to complete if zoomed out to a smaller scale.

9.3.6 Identify

Tapping on the identify button (6) will activate a function for identifying objects in the map and retrieving information about them. Once the function is activated, attributes are fetched by tapping on objects in the map. Most of the objects in the layers are queryable (see full layer declaration in Table 9.1).

9.3.7 Select

Selecting drawn vector features is done by activating the select feature (7). This will display information in a pop up about the feature and the note attribute bound to it.

9.3.8 Write note

When a feature is selected and this button (8) is tapped, a prompt appears where a note can be written for the feature. A note will appear as the Notering attribute on the feature pop up when it is selected and directly in the map if the Noteringar layer is active. If no feature is selected, tapping this button will add a note to the feature that was last drawn.

9.3.9 Help

Tapping this button (9) opens up the Swedish user manual in a new browser tab. The manual is more thoroughly presented in chapter 9.5 User Manual and the full manual is found in the Appendix - Swedish Manual.

9.3.10 Status field

This field (10) shows messages about current activities. For example it displays the function that is currently in use.

9.3.11 Color switcher

Tapping this button (11) generates a popup window where the user can select which color to draw features in (see figure 9.4). The current color is displayed in text on the button.

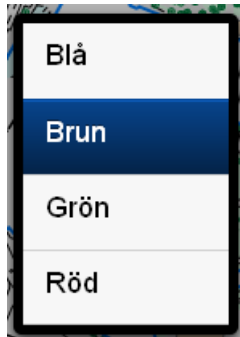


Figure 9.4: *Color selection list.*

9.3.12 Remove feature

Tapping this button (12) enables the remove function. If active, tapping on a drawn feature on the map removes it.

9.3.13 Modify feature

Tapping this button (13) enables the modify feature function. Tapping a feature allows its vertices to be modified and its position to be changed. When modifying features of different colors the modify button has to be tapped every time the user wants to modify a feature of different color.

9.3.14 Draw point

Tapping this button (14) allows the user to draw a point feature on the map by tapping the map at the position where she wants to draw.

9.3.15 Draw line

Tapping this button (15) allows the user to draw a line feature on the map by tapping the map at the position where she desires vertices of the line to be placed. The user finishes drawing by double tapping on the map.

9.3.16 Draw polygon

Tapping this button (16) allows the user to draw a polygon feature on the map by tapping the map at the position where she desires vertices of the polygon to be placed. The user finishes drawing by double tapping on the map.

9.4 General functionality

When working with the application there is some general functionality that performs tasks in the background. This makes the application more user-friendly and reduces the risk of application failure due to user mistakes.

An example of this is when a new function is activated, all other functions are deactivated and a text appears in the status field to notify the user that his action was performed. Another example is when using the identify function on point layers. A point on the map is very hard to hit with a tap of the finger. Here the hit boxes have been enlarged to make this easier. A third example is when the notification function is activated and no drawn feature is currently selected. Normally the selected feature will be the feature the note will be attached to, but if no selection is made the last drawn feature will get the written note attached. It is clearly shown what feature the notification gets bound to. When done notifying a feature the select function is automatically activated so that the user can continue selecting features to notify.

9.5 User Manual

This chapter is only a short description of the user manual and of what it contains and handles. The full user manual is found in Appendix B. The manual is built up by a docked menu panel on the left side and a page for showing the selected information on the center and right of the screen (see figure 9.5).

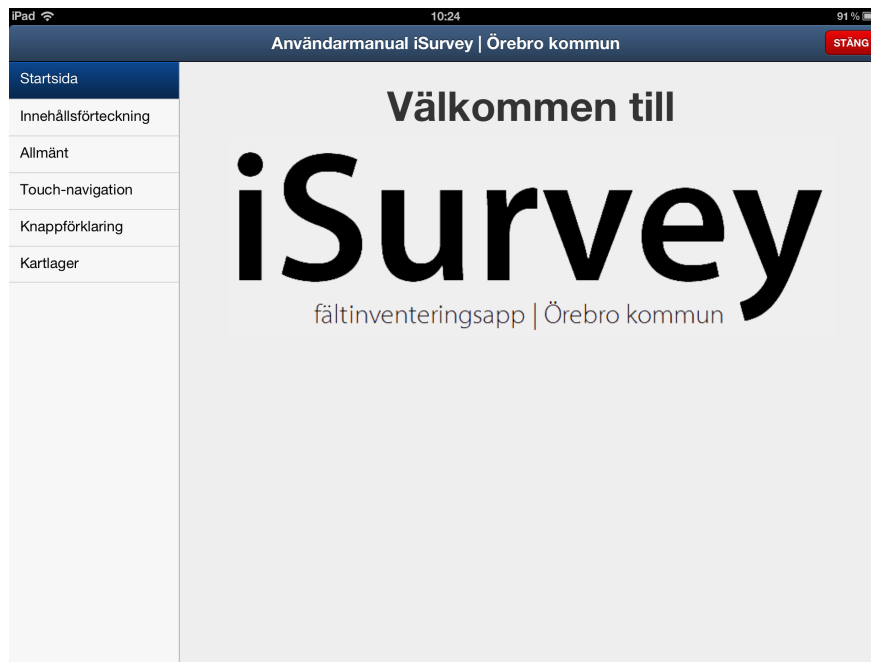


Figure 9.5: *User manual.*

The menu contains a starting page, a table of contents, general information, an explanation of touch gestures, a toolbar guide and a list of available map layers. When the user is done using the manual she simply taps the STÄNG button in the top right corner to return to the map.

9.6 Map layers

A more thorough view of the layers that are included in *iSurvey* is presented in Table 9.1. It contains all available layers, presented in its left-most column. The middle column shows what label in the layer switcher the layer is grouped under. In the right-most column an identifiable layer is presented with a Yes or as a list if more than one.

Table 9.1: *Layers in the layer switcher. Baskarta and Ortofoto are the two base layers. The rest of the layers are viewed on top of these two layers.*

Layer	Layer switcher label	Identifiable layer
Land use	Baskarta	No
Roads	Baskarta	Yes
Railroads	Baskarta	Yes
Buildings	Baskarta	Yes
Building details	Baskarta	Yes
Street boundaries	Baskarta	Yes
Water boundaries	Baskarta	Yes
Properties	Baskarta	Yes
Property boundaries	Baskarta	Yes
District boundaries	Baskarta	Yes
Addresses	Baskarta	No
Street names	Baskarta	No
District names	Baskarta	No
Trees	Baskarta	Yes
Ortophoto	Ortofoto	Buildings, building details, properties
Proposed borders	Gränser	Yes
Property border points	Gränser	Yes
Property border point numbers	Gränser	Yes
Temporary surveyor text	Mätttext	No
Reference points	Stompunkter	Yes
Height reference points	Stompunkter	Yes
Objects surveyed in detail	Övriga objekt	Yes
Property rights	Rättigheter/ bestämmelser	Yes
Regulations	Rättigheter/ bestämmelser	Yes
Plans	Planer	Yes
Water pipes	VA-ledningar	Yes
Surface and storm water pipes	VA-ledningar	Yes
Waste water pipes	VA-ledningar	Yes
Note attributes for drawn features	Noteringar	No

9.6.1 Changing the content of the map

What is shown on the map is controlled by what is present on the server side of the application. That information is fetched to its full extent, both when it comes to available layers and their attributes. This means changing available layers and their attributes can be done server side, without changing any code within the application. The information can therefore be tailored after the present needs of the units using it.

9.7 Examples of iSurvey use

In this section, three user examples are presented, each of them relevant to a certain municipal unit. The examples are meant to explain how *iSurvey* can be used by a surveyor, a land surveyor and an employee of the environmental unit in a typical field work situation. The chosen examples are:

1. Surveying inventory
2. Land surveying field work
3. Sewer pipe plotting and examination

All cases are fictional examples of how the application *iSurvey* might be used in a real situation. The numbers in the text refer to what button is tapped to activate a function.

9.7.1 Example 1 - Surveying inventory

Conducting field inventory is supposed to lay ground for what is to be surveyed in the future to keep the maps up to date. A surveyor conducting a field inventory is a very clear example of when and how this application can be used. The inventory area is usually a whole block of properties, but in this example only a few properties are in focus. Since the application map is dynamic in what extent, zoom level and visible layers are shown, this makes no difference in how the work is conducted.

Case - New building, building details and complementary building on Örebro Citronen 2 and Citronen 3 The surveyor is in a newly developed residential area where several buildings have been built recently. The job to be conducted is to inventory how much surveying work is needed in this area and add it to the GIS-program in the office. For this, *iSurvey* will be used. The work flow description below concerns the inventory of new buildings on Örebro Citronen 3 and new building detail on Örebro Citronen 2.

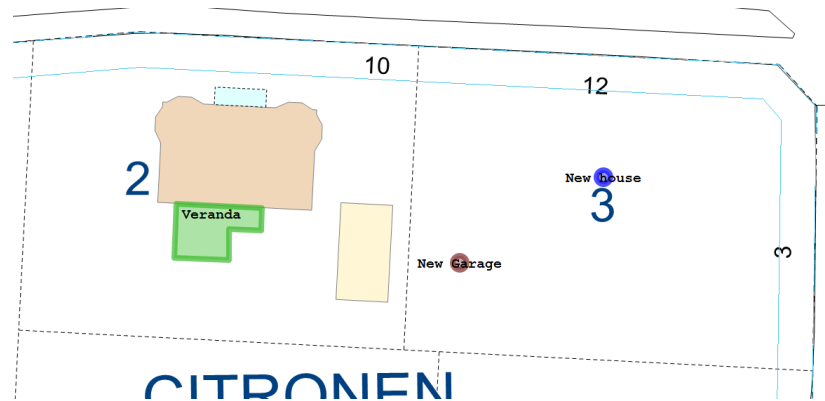


Figure 9.6: An example of drawn features to be exported to the database. The solid lines represent plan borders.

Workflow The area of interest is situated near the center of Örebro, and therefore there is no need to cache the map extent before leaving the office. When in the field and on the property in question the surveyor taps the geolocation button (2) and zooms to an appropriate zoom level (pinching). There are two objects of interest to the surveyor on Citronen 3, a new building and a new complementary building (a garage). On Citronen 2, there is a new building detail built on the house (a veranda). The surveyor decides appropriate colors (11) for the different types of buildings, and draws them as point objects (14) (see Figure 9.6). The veranda is drawn in more detail as a polygon feature (16) to conform to its real shape. Next, notes are written labelling the buildings as house, garage and veranda (6) (7). The Layer switcher (1) is used to add the layers Planer, Rättigheter/bestämmelser and Noteringar to estimate if the buildings are overlapping areas that are not allowed to be built on. This will not give the surveyor an exact location of the house in accordance with the plan, but enough to know if further investigation is needed. The notes now show in the map. The inventory for this property is finished and the drawn features are exported to the database (4).

9.7.2 Example 2 - Land surveying field work

For most land surveying cases there is field work needed. Usually a meeting is held with all parties that are affected in an errand. In these situations questions are asked that may be hard to answer without using the office computer and its GIS-programs. In the field, land surveyors almost always need to use the surveying equipment. The main reason for bringing a map is for handling as much information as possible. A limited amount of notes and features is drawn in the next case; it concentrates on using *iSurvey* for viewing the map and querying objects.

Case - Land surveyor meeting with affected parties in field A land surveyor is meeting the affected parties in field, regarding a re-allotment between two properties.

For this, *iSurvey* is used for viewing and querying the map to show the affected parties how something on the map looks like (e.g property boundaries) and to get information about buildings and properties in the area. The re-allotment concerns an area of approximately 4,000 square meters that is to be regulated from Örebro Ervalla 1:39 to Örebro Ervalla 2:15.

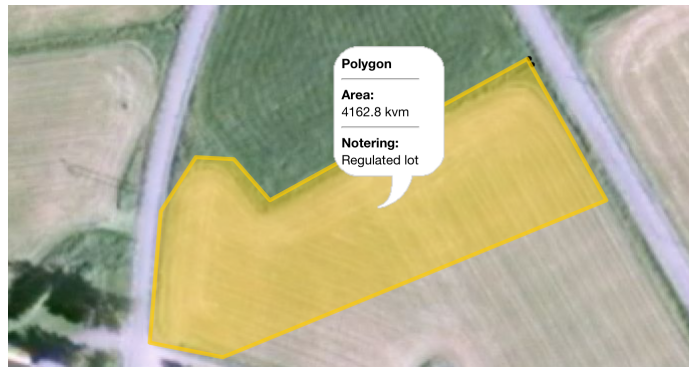


Figure 9.7: Area to be regulated originally drawn on top of the orthophoto.

Workflow Before leaving the office, a search is made for the property Örebro Ervalla 1:39 (3). The surveyor suspects the internet connection will be weak, and therefore caches (5) the extent needed to be viewed, for both Baskarta and Ortofoto. The features that are most important are probably identify and the WMS layers, which need internet to run, but when the internet connection is weak it is good to cache as much as possible. An area of 4,000 square meters is to be re-allotted. For this an example area is drawn (16) on top the orthophoto (1) and given a note (8). The area is then adjusted (13) to the property borders in the base map (1) (see figure 9.7 and 9.8) for a better fit. The property border point layer is added to the map (1), and queries are made (6) on border points, properties, property borders and buildings to answer questions from the affected parties.

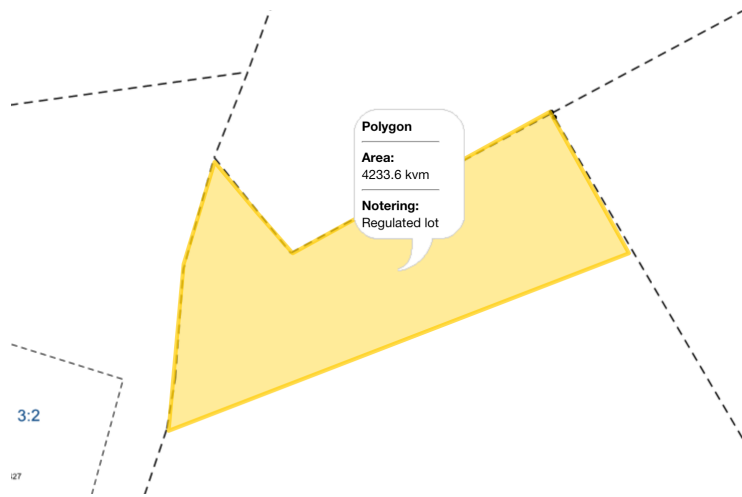


Figure 9.8: Area to be re-allotted, fitted to match property borders.

9.7.3 Example 3 - Environmental unit field work

The environmental unit conducts field work to examine, inspect and plot new and old sewer pipes. An inspection should result in a sewage facility drawn on a map, and a filled out inspection form with reference to objects and positions on the map. In the following example the form is filled out as usual with references made to drawn features in *iSurvey*.

Case - Sewer pipe plotting and examination on Örebro Olaus Petri 3:229 A new sewage facility has been constructed on Örebro Olaus Petri 3:229. An employee from the environmental unit will conduct field work on this property, fill out the inspection form and use *iSurvey* as a map.

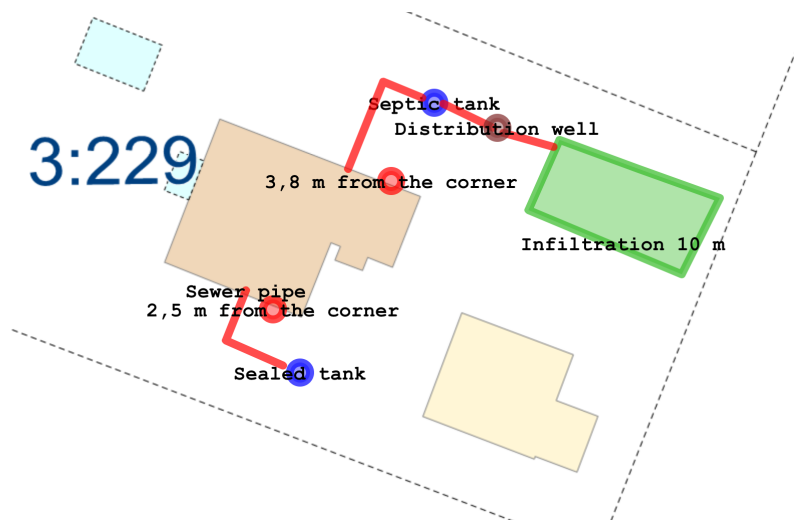


Figure 9.9: *Infiltration facility inventory example.*

Workflow The property is situated close to the city center and therefore the internet connection is predicted to be good and no caching is needed. Upon arrival at the property, the geolocation button (2) is used, along with pinching to get an appropriate zoom level. The resident shows on the ground where the pipes are located and they are drawn (15) accordingly. The sealed tank, the septic tank and the distribution well is also located and mapped with point features (14). The 10 m long infiltration facility is then mapped as a polygon (16) (see figure 9.9). All these features in the map are given notes describing what they are (8). It is important to note how far from the house corner the pipes are attached. For this point features are drawn (14) and noted (8). Features are then exported to the database (5).

9.8 License

The open source libraries that are integrated in the implementation of *iSurvey* are *OpenLayers*, *Sencha Touch* and to a small extent *jQuery mobile*. *OpenLayers* uses a *FreeBSD* license and *jQuery mobile* uses an *MIT* license. These two very unregulated licenses are both compatible with *GNU GPL* license that *Sencha Touch* uses (GNU, 2012). What these licenses state is that the source code for any application developed using software under these licenses, has to remain open. This is the case with *iSurvey*, since all code can be reached from any modern web browser. Additionally, the application can not be further licensed under another stricter license. This is not the case with *iSurvey*. The scripts included in the application are marked to show they are bound under these licenses.

The application was developed by the authors while assuming a position as employees

at *Örebro municipality*, which means the authors does not "own" the application. Since it will not take a role as a proprietary software and since it is developed under such free licenses, ownership does not matter. Anyone can retrieve, modify and reuse the source code.

10 Discussion

During the process of developing the *iSurvey* application and writing of this thesis several problems were met along with associations to other fields. In this chapter, the topics dealt with throughout this master thesis are discussed.

When brainstorming functionality and design for *iSurvey* the authors looked into current software to gather information and ideas. The application Nya Hemnet had a large impact on the authors' hopes of how *iSurvey* might end up being developed and designed. Nya Hemnet has a free hand drawing function that locks the map in place and allows the user to drag the finger across the screen and draw a polygon. This feature was not added to *iSurvey* due to two reasons: 1) drawing with free hand technique resulted in polygons with a cumbersome amount of vertices that caused problems for the export function, and 2) locking and unlocking the map when out in the field may lead to unwanted and unnecessary amounts of button pressing in order to draw. The design of Nya Hemnet was also very appealing to the authors, but given the limited time for development and the complexity of implementing a similar design, this idea was not taken into further consideration.

Because of the relatively new field of mobile map applications, not much research has been done on interaction design and usability for it. Further investigation in the field is needed as location based services and map applications are used more and more. The research done however is an adaptation of existing usability and interaction design heuristics and principles from a general point of view onto a map point of view. It is important for developers of map applications to understand these heuristics and principles because of the potentially large amount of users their applications can reach.

There are a vast number of technologies to choose from when developing both native and mobile web applications, and describing them all would be almost impossible. The selections made in this thesis were all either considered by the authors at some point in time, or used in the final product. The authors find it fascinating how well the technologies can be integrated with each other and be brought together to form a single unified product. That is the power of using open source libraries and web standards for, in this case, web GIS.

The overall process of designing and developing the application was performed with a user-centered approach. The first stage of the implementation of *iSurvey* consisted of the authors having meetings with future end-users to discuss their thoughts and needs concerning functionality. Design was not discussed in great depth during these meetings. The use of prototyping was also never properly utilized. This was mainly because the current layout is standardized by open source tools. The users who were interviewed can be seen as representative users of their target group. Because of this, their needs were considered true and general for each of their respective groups and further interviews were deemed unnecessary as well as time and resource consuming. These meetings lead to an understanding of the users and their needs. They could then be transcribed into requirements. It is from these requirements that implementation of

iSurvey began. Throughout the entire process constant showcasing of current progress was done for the end-users who could then give feedback on the product. This was to enforce an iterative process. All negative/constructive feedback led to updates of the application to better fit the users' needs. The product was finished when requirements were met to the extent that the users were satisfied with the application. This process can be related to the interaction design process, which is essential for creating products that will be well received by its users.

The requirement specification is general and only concerns the requirements on functionality. Design requirements were left out of the requirement specification because of the way open source software comes with predefined layouts and color schemes that the authors and end users considered sufficient enough for the purpose of the application. Had the requirement specification included requirements for design, they would concern things as toolbar placement, icon design, hex coding for color schemes etc. No requirements were put on the cartography of the application. The current cartography design at *Örebro municipality* is used in the base map of the application. It is also used in other maps, for example on the municipality website. This cartography design makes it easier for novice map users to understand the information in the map while still being adequate enough for expert users to use in field work.

10.1 Dismissed requirements

Not all requirements were met and included in the application. Below is a list of dismissed requirements followed by an explanation of why the requirements were dismissed.

- SR2.1 Allow for photos to be taken when out in the field.
 - SR2.1.1 Taken photos should have a date stamp.
 - SR2.1.2 Photos should be geotagged.
- SR2.3 Tracking and logging current position (as a point).
 - SR2.3.1 The GPS accuracy at the moment of tracking and logging should be saved.
- SR2.7 Include different profiles for the different units at the department.
- SR3.2 Tracking and logging line segments.
 - SR3.2.1 The GPS accuracy at the moment of tracking and logging should be saved.

Since a web application is developed, the application cannot access a device's camera. This means that the requirements SR2.1, SR2.1.1 and SR2.1.2 are impossible to implement into a web application. This sort of functionality requires a native application to be developed or integrated to work as an importing application on the side.

The built in GPS receiver uses C/A-code pseudorange measurements and has a mean error of around 5-10 meters at best (Klang and Ågren, 2007). Additional sources of errors make this measuring accuracy even worse. This means that SR2.3, SR2.3.1, SR3.2 and SR3.2.1 are not accomplishable to the final use of the application by the developers and the department. The purpose is to log the user's position with reliable accuracy, but since this cannot be achieved these requirements are dismissed. The Geolocation function works in a similar way, but it is not used for getting a reliable, accurate position that can be logged and exported, but more for quick orientation and finding the general position of the user. Minor informal field tests of device GPS accuracy has been conducted, which underpin the decision to dismiss these needs.

SR2.7 became superfluous after discussions between the developers and the municipal department. When the requirements were first established, the interface had not been developed. The developers were afraid that the interface may become cluttered if all the functionality was to be packed into it at once. This made the idea of different profiles for the different departments interesting, because some departments have use for certain functionality while not needing other functionality at all. However, the interface did not become cluttered when including all of the functionality, so this requirement is discarded. The choice of different colors while drawing features in the map was successfully implemented, which also contributes to the decision to not implement different profiles.

Some of the dismissed needs, especially the dismissed needs of being able to take geographically bound photographs will drastically reduce the applications usability for the Planning permissions unit (Bygglov), because of the nature of their field work which requires photographs to be taken. There is however already a mobile application at *Örebro municipality* that extracts the position of a photograph from its geotag-information so it can be displayed on a map where the photo was taken.

In the application there are a few operations that were implemented by someone else and not by the authors of this thesis. These are the search and identify functions. These features were implementing by the Geographical IT department at Sweco in Örebro. The authors have however slightly modified some parts of these functions to better fit the needs of the applications. A way of securely logging in to the application was a function that was not in the requirement specification, but has been added after the requirements were met and the authors' work was completed. The secure log in was also implemented by the department mentioned above, but after the handover of the project.

10.2 Further development

Further development of the application may need to be done in the future to satisfy needs that may arise. These could include adding additional layers to the map and changing attributes that are displayed after using the identify function. The authors have reviewed the code together with the head of the *Map and surveying unit* at *Örebro municipality* to provide him with an understanding of how additional layers can be added through modifying bits of code.

The attributes that are fetched when using the identify function are handled by the *Map and surveying unit*, so it is their responsibility to include what they regard as necessary. There is no need to alter any application code to fix this, because of the way the parser of the identify results is implemented. Currently, using the identify button on a property will result in a link, among other things, to the cadastral system of *Örebro municipality*. However, this system cannot be accessed by anything other than the intranet at *Örebro municipality*, i.e. not even from within the demilitarized zone (DMZ). It could be an interesting addition to add a safe access to this cadastral system.

The cartography design of the application can in the future be changed as well if it would show that the current design does not satisfy the needs of the users. This would not require any modification of the application code and is handled by the *Map and surveying unit*.

Currently it is in the works to bring device specific functionality to web applications by using JavaScript APIs together with HTML5. If and when this gets implemented, it would of course be a welcomed addition to *iSurvey*. Being able to access the camera and take photos directly from the application would allow *iSurvey* to serve an even greater set of needs than its current implementation. This would mean that some of the dismissed needs mentioned above could get implemented and allow for example the application to be used by the planning permissions unit to a greater extent.

In its current state using *iSurvey* outside of *Örebro municipality* is not possible, at least not if one wishes to use it for its intended purpose. This is because only data from *Örebro municipality* is available in the application. One can however see *iSurvey* as a shell for these types of applications, because all of the functionality is independent from the actual map. Say Lund municipality wishes to use an application with similar functionality to *iSurvey*'s. If Lund has a comparable system for storing geographic data (TileCache and/or WMS vector data and servers) then it would only be a matter of changing a few lines of code (like adding correct URLs, names of layers, and correct coordinates for the bounds of the map etc) for the application to be able to fetch and display Lund municipality's data. If the data storage and structure is different, there would have to be a greater change in the code that concerns fetching data, or the needed TMS and WMS servers would have to be constructed.

10.3 Future technique - augmented reality

An interesting field in the future of GIS applications is the use of augmented reality (AR). If you at one end of the spectrum have virtual reality (VR) and at the other end reality itself, AR would be classified somewhere in between. The basic idea of it is adding virtual objects in real-time to reality. The term 'augment' means to make something better, which in AR's case is to make reality better. Currently several AR projects are being worked on, and the solutions mostly circulate around using a pair of glasses that can have graphics displayed on them. This would allow the user to for example see their walking path to a destination when using a navigation application or to see

information about what hours a store is open and what deals they offer right now, just by looking at it.

This can lead to a spectacular development in GIS applications for AR. The context of use and usability guidelines will be completely redefined by having things projected on to reality instead of having them on a screen.

AR could be used for a similar purpose compared to *iSurvey's* by for example virtually drawing an area on the ground and allowing the surveyor to view it in real time at its actual location out in the field. This may seem like a dream, but it may be available in the not too distant future.

11 Conclusions

The general purpose of the application *iSurvey* was to reduce the amount of time used for preparation of field work, and to further digitalize the working environment. This purpose, especially the part of saving time for preparation, has started to be fulfilled. It is however a longer process than just starting to use this application. Users also have to become more acquainted with it over time, for this purpose to be achieved fully (Nordström, 2012).

Another part of the general purpose was developing a tool for conducting field inventory and to replace paper maps. This purpose has been successfully fulfilled for conducting certain tasks, while other tasks are still more manageable with old procedures (Nordström, 2012).

Implementing and designing a mobile map application that fulfills the needs and requirements of departments in a Swedish municipality, was overall successfully achieved.

The purpose was divided into partial purpose definitions. The first was conducting a needs analysis that was to result in a requirement specification. This purpose was effectively achieved, a needs analysis and requirement specification were produced. As mentioned in 10 Discussion, several steps of a development process that were treated in chapters 3 Interaction design and 4 Advanced interaction design in mobile GIS, were left out in the *iSurvey* development. Had these steps been considered at this stage of the development process, the outcome might have been different.

The broad topics of interaction design and technical solutions available for mobile map development were to be reviewed. This purpose was successfully achieved and presented in this master thesis. It was discovered that the topic of the interactive design process for map applications was not immensely investigated. There was however some available research that helped in adapting the general interaction design material to a mobile map utilization. There are on the contrary a large amount of available techniques for mobile map application development. Relevant parts of this very broad topic were selected, investigated and presented successfully in this thesis.

References

Android Bandhana (2012), *digital image of Touch gestures*, available at <http://androidandsandu.blogspot.se/2012/05/multi-touch-pinch-zoom-support-to.html>, viewed 2012-07-25.

Android SDK (2012), Android developers official web site, available at <http://developer.android.com/sdk>, viewed 2012-09-13.

Apple Developer (2012), Apple Developer official web site, available at <https://developer.apple.com/technologies/>, viewed 2012-09-13.

Arnberg W and Rystedt B (2008), *Kapitel 11, Kartografi, p 280*, In: Harrie L (editor), *Geografisk informationsbehandling, Teori, metoder och tillämpningar, fjärde upplagan* Stockholm, Sweden: Forskningsrådet Formas och författarna.

Avloppsappen (2012), "Var ligger avloppet?", available at http://www.lansstyrelsen.se/vastmanland/sv/miljo-och-klimat/verksamheter-med-miljopaverkan/tillsynsvagledning/Pages/enskilda_avlopp.aspx, viewed 2012-09-14.

Bagnall J and Koberg D (2003), *The Universal Traveler: a Soft-systems Guide: to Creativity, Problem-solving, and the Process of Design*, Michigan, USA: Crisp Learning.

Brimicombe A and Li C (2009), *Location-based Services and Geo-Information Engineering, pp 67-108*, Chichester, UK: John Wiley & Sons Ltd.

Burleson Consulting (2012), *Relational Database Objects and Abstract Data Types*, available at http://www.dba-oracle.com/t_object_relational_data_types.htm, viewed 2012-09-26.

Card S, Moran T and Newell A (1986), *The Psychology of Human-Computer Interaction (New edition)*, Hillsdale, USA: Lawrence Erlbaum Associates.

Coronel, Morris and Rob (2010), *Database Systems - Design, Implementation, and Management (9th Edition)*, Boston, USA: Cengage Learning.

Dundas Data Visualization (2012), *digital image of map simplification*, available at <http://support2.dundas.com/OnlineDocumentation/WebMap2005/Images/MapSimplification1.bmp>, viewed 2012-10-09.

ESRI (1996), Environmental Systems Research Institute, *Introduction to Map Design*.

Evjen B, Nagel C and Stephens R (2010), *C# 4, ASP.NET 4, and WPF, with Visual Studio 2010 Jump Start*, Indianapolis, USA: Wiley publishing.

Freeman E and Robson E (2011), *Head first HTML5 programming*, Sebastopol, USA: O'reilly Media.

Garrett J (2005), *Ajax: A New Approach to Web Applications*, available at www.adaptivepath.com/ideas/ajax-new-approach-web-applications, viewed 2012-09-18.

GIYF (2012), Google Is Your Friend, available at <http://www.google.com>

GNU (2012), GNU official web site on GPLv3, available at <http://www.gnu.org/licenses/quick-guide-gplv3.html>, viewed 2012-09-25.

Gollman D (2011), *Computer Security, third edition, pp 156-176, pp 319-338*, Chichester, UK: John Wiley & Sons Ltd.

GoMo News (2012), *digital image of navigation application*, available at <http://www.gomonews.com/wp-content/uploads/2010/01/image001.png>, viewed 2012-10-09.

HaptiMap (2012), Magnusson C (editor), Rassmus-Gröhn K (editor), Deaner E (editor), *User requirements and design guidelines for map applications*.

Harold (2004), Harold E R, *Java Network Programming, 3rd edition*, Sebastopol, USA: O'Reilly Media.

Helander M, Landauer T and Prabhu P (1997), *Handbook of Human-Computer Interaction 2nd Edition*, The Netherlands: Elsevier Science BV.

Hemnet (2012), Hemnet official web site, available at www.hemnet.se, viewed 2012-09-13. ISO (2012), International Organization of Standardization official web site, available at <http://www.iso.org/>, viewed 2012-09-25.

Jalote P (2008), *A Concise Introduction to Software Engineering*, London, UK: Springer-Verlag London Limited.

Jenny B, Jenny H and Räber S (2008), *Chapter 3 Map design for the Internet*, In: Peterson, M. P. (editor), *International Perspectives on Maps and the Internet*, New York, USA: Springer-Verlag Berlin Heidelberg.

jQuery mobile (2012), *Mobile Graded Browser Support*, available at <http://jquerymobile.com/gbs/>, viewed 2012-09-13.

Klang D and Ågren J (2008), *Kapitel 5, Insamling av geografiska data, pp 90-115* In: Harrie L (editor), *Geografisk informationsbehandling, Teori, metoder och tillämpningar, fjärde upplagan*, Stockholm, Sweden: Forskningsrådet Formas och författarna.

Leaflet (2012), Leaflet official web site, available at <http://leaflet.cloudmade.com>, viewed 2012-09-14.

Maso J, Pons X and Singh R (2010), *OGC WMTS and OSGeo TMS standards: motivations, history and differences*.

Microsoft (2012), Microsoft ASP.NET official web site, available at www.asp.net, viewed 2012-09-18.

Mobithinking (2010), *Mobile applications: native v Web apps - what are the pros and cons*, available at <http://mobithinking.com/native-or-web-app> , viewed 2012-09-12.

Morris R (2009), *The Fundamentals of Product Design*, Lausanne, Switzerland: AVA Publishing SA

Nielsen, J (1994), *Usability Engineering*, San Diego: Academic Press.

Nivala A-M and Sarjakoski L-T (2005a), *Adaptation to Context – A way to Improve the Usability of Maps*.

Nivala A-M, Sarjakoski L-T and Sarjakoski T (2005b), *User-centered Design and Development of a Mobile Map Service*.

Nivala A-M, Sarjakoski L-T and Sarjakoski T (2007), *Usability Methods' Familiarity among Map Application Developers*.

Norman D (1988), *Design of everyday things*, New York, USA: Basic Books inc.

Nya Hemnet (2012), iOS application Nya Hemnet.

OGC (2005a), Open Geospatial Consortium, *OpenGIS Web Map Service Implementation specification*.

OGC (2005b), Open Geospatial Consortium, *Web Feature Service Implementation specification*.

OGC (2010), Open Geospatial Consortium, *OpenGIS Web Map Tile Service Implementation specification*.

OGC (2010), Open Geospatial Consortium, *OpenGIS Implementation Specification for Geographic Information - Simple feature access - Part 1: Common Architecture*.

OGC (2012), Open Geospatial Consortium official web site, available at <http://www.opengeospatial.org>, viewed 2012-09-25.

Open Source Initiative (2012a), Open Source Initiative on *MIT License*, available at <http://opensource.org/licenses/MIT>, viewed 2012-09-25.

Open Source Initiative (2012b), Open Source Initiative on *BSD Licenses*, available at <http://opensource.org/licenses/bsd-license.php>, viewed 2012-09-25.

OpenLayers (2012a), OpenLayers Examples, available at <http://openlayers.org/dev/examples>, viewed 2012-09-10.

OpenLayers (2012b), OpenLayers official web site, available at <http://openlayers.org>, viewed 2012-09-13.

OSGeo (2012a), Open Source Geospatial Foundation on *OpenLayers*, available at <http://trac.osgeo.org/openlayers/>, viewed 2012-09-13.

OSGeo (2012b), Open Source Geospatial Foundation wiki on *Tile map service specification*, available at http://wiki.osgeo.org/wiki/tile_map_service_specification, viewed 2012-09-14.

OSGeo (2012c), Open Source Geospatial Foundation on *Proj.4*, available at <http://trac.osgeo.org/proj/>, viewed 2012-09-16.

OSGeo (2012d), Open Source Geospatial Foundation on *proj4js*, available at <http://trac.osgeo.org/proj4js/>, viewed 2012-09-16.

Preece, Rogers and Sharp (2007), *Interaction design, beyond human-computer interaction*, Barcelona, Spain: John Wiley and Son Ltd.

Runkeeper (2012a), Runkeeper official web site, available at www.runkeeper.com, viewed 2012-09-13.

Runkeeper (2012b), iOS application Runkeeper.

Sencha (2012), Sencha official web site, available at <http://www.sencha.com>, viewed 2012-09-14.

Shneiderman B, Plaisant C, Cohen M and Jacobs S (2009), *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 5th edition, Boston, USA: Addison-Wesley.

SQA (2001), Software Quality Assurance ISO standard, available at <http://www.sqa.net/iso9126.html>, viewed 2012-10-02.

Techrepublic (2012), *digital image of DMZ networking*, available at <http://techrepublic.com.com/i/tr/cms/contentpics/5756029-DMZ-overview-A.gif>, viewed 2012-09-17.

TileCache (2012), TileCache official web site, available at <http://tilecache.org>, viewed 2012-09-14.

UseIT (2005), Jakob Nielsen's website on the ten usability heuristics, available at http://www.useit.com/papers/heuristic/heuristic_list.html, viewed 2012-10-09.

W3C (2012a), World Wide Web Consortium on Web design standards, available at www.w3.org/standards/webdesign/htmlcss, viewed 2012-09-18.

W3C (2012b), World Wide Web Consortium on HTML5, available at <http://dev.w3.org/html5/spec/>, viewed 2012-09-25.

W3Schools (2012), W3Schools on XML, available at <http://www.w3schools.com/xml/>, viewed 2012-09-18.

WebGL Earth (2012), *digital image of a Mercator tile pyramid*, available at <http://data.webglearth.com/doc/webgl-earth2x.png>, viewed 2012-09-14.

WebKit (2012a), WebKit official website, available at <http://www.webkit.org/>, viewed 2012-09-25.

WebKit (2012b), WebKit wiki, available at <http://trac.webkit.org/wiki>, viewed 2012-09-25.

Zwick (2010), *Sencha Touch: The HTML5 Mobile App Framework*, available at <http://mobile.tutsplus.com/articles/news/sencha-touch-html5-mobile-framework/>, viewed 2012-09-13.

Personal communication

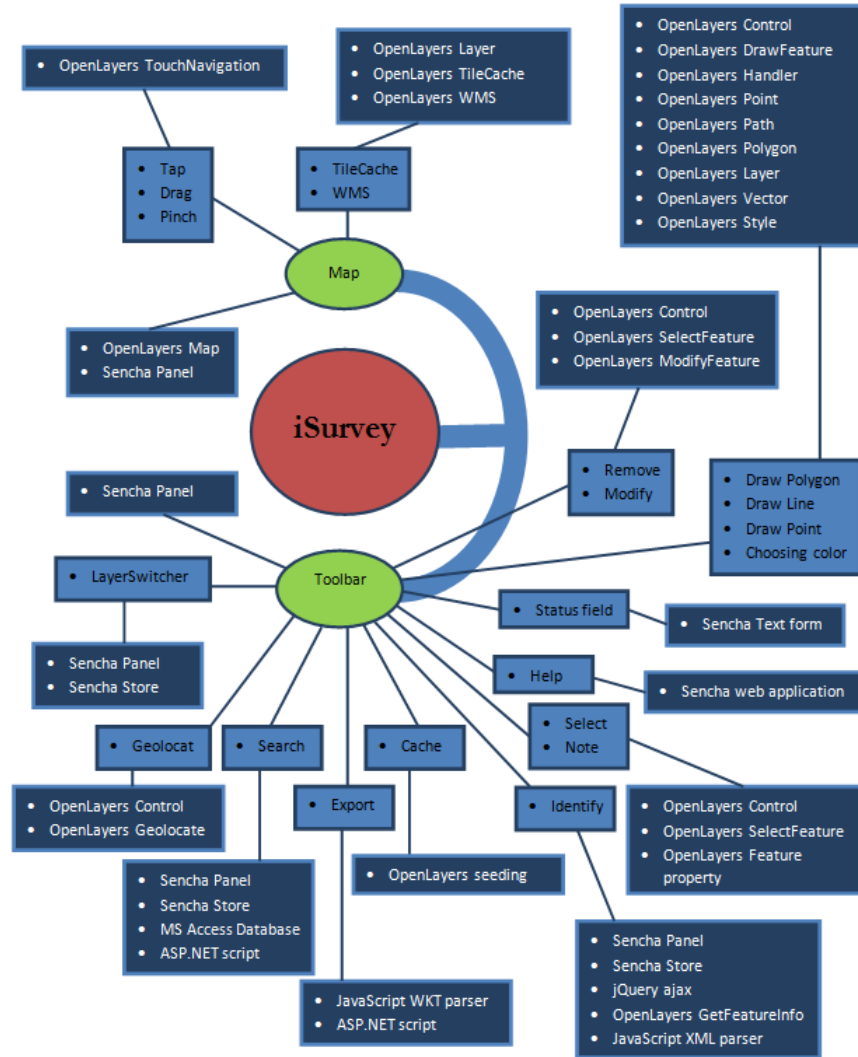
Bergström (2012), Dennis Bergström at Sweco Positions Örebro, continuous contact 2012-06 - 2012-08.

Nordström (2012), Daniel Nordström, Head of Kart- och mätenhet Örebro kommun, continuous contact 2012-06 - 2012-11.

Peterson (2012), Stefan Peterson, Head of Lanmäteri- och geodataavdelning Örebro Kommun, continuous contact 2012-06 - 2012-08.

Eriksson (2012), Joakim Eriksson, TeknD and researcher at Ergonomics and Aerosol Technology LTH, guest lecture 2012-09-19.

Appendix A - iSurvey figure



Appendix B - Swedish iSurvey manual

In this appendix the Swedish manual is presented. This manual is the same as the manual shown in the app when the question mark button is tapped. It is designed to be a simple-to-understand, non-technical manual which aims to break down usage into the most basic parts.



Innehållsförteckning

1 Allmänt	i
1.1 Referenssystem	i
1.2 Licenser	i
1.2.1 Sencha Touch	i
1.2.2 OpenLayers	i
2 Touch-navigation	ii
2.1 Grundläggande Touch-gester	ii
2.2 Navigera	iii
2.3 Zooma in	iii
2.4 Zooma ut	iii
3 Knappförklaring	iv
3.1 Lager	iv
3.1.1 Baskartor	v
3.1.2 Övriga lager	v
3.2 Geolocate	v
3.3 Sök	vi
3.4 Export	vii
3.5 Lagring av kartvy	vii
3.6 Identifiera	viii
3.7 Markera	viii
3.8 Notera	ix
3.9 Hjälp	x
3.10 Statusfält	x
3.11 Rita punkt	xi
3.12 Rita linje	xi
3.13 Rita polygon	xi
3.14 Färg	xii
3.15 Ta bort objekt	xiii
3.16 Ändra objekt	xiii
3.16.1 Ändra punkt	xiii
3.16.2 Ändra linje	xiv
3.16.3 Ändra polygon	xv
4 Kartlager	xvi
4.1 Baskarta	xvi
4.2 Ortofoto	xvi
4.3 Gränser	xvi
4.4 Mättext	xvii
4.5 Stompunkter	xvii
4.6 Övriga objekt	xvii
4.7 Rättigheter/bestämmelser	xvii
4.8 Planer	xvii

4.9 VA-ledningar	xvii
4.10 Noteringar	xvii

1 Allmänt

1.1 Referenssystem

Kartan har SWEREF 99 15 00 som referenssystem.

1.2 Licenser

Då open source-kod har använts vid utvecklingen av applikationen är de olika utvecklingsverktygens licenser högtintressanta.

1.2.1 Sencha Touch

Sencha Touch har en egen gratis Commercial License vilket innebär att iSurvey får utvecklas i Sencha Touch SDK utan kostnad.

1.2.2 OpenLayers

OpenLayers använder sig av en FreeBSD-licens vilket innebär att OpenLayers får användas gratis så länge man har med licensavtalet vid redistribuering av käll- och binärkod.

2 Touch-navigation

2.1 Grundläggande Touch-gester



Figure 1: Trycka. Bilden hämtad från *Android Bandhana* (2012) och sedan modifierad

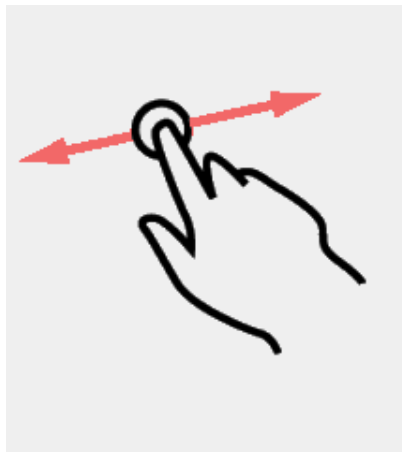


Figure 2: Dra. Bilden hämtad från *Android Bandhana* (2012) och sedan modifierad

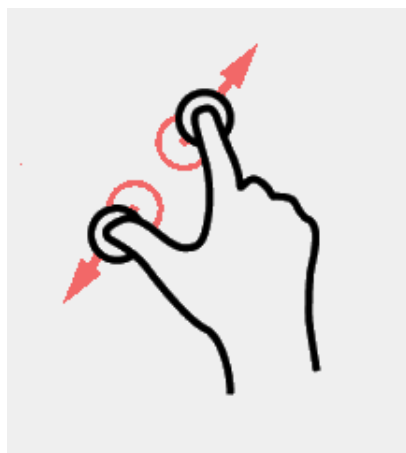


Figure 3: "Pincha", för tummen och pekfinger mot eller ifrån varandra. Bilden hämtad från *Android Bandhana* (2012) och sedan modifierad

2.2 Navigera

För att röra sig på kartan så trycker du ner fingret mot skärmen och håller det nere. Dra sedan fingret över skärmen för att förflytta dig i kartan (*se figur A.1 och A.2*).

2.3 Zooma in

För att zooma in på kartan så trycker du ner tummen och pekfinger på skärmen och för dem ifrån varandra. Du kan även trycka två gånger snabbt på skärmen för att zooma in på det ställe du tryckte på (*se figur A.3*).

2.4 Zooma ut

För att zooma ut på kartan så trycker du ner tummen och pekfinger på skärmen och för dem mot varandra (*se figur A.3*).

3 Knappförklaring

3.1 Lager



Figure 4: *Välja lager*

När du trycker på denna knapp så presenteras en lista på skärmen med ett antal lager som kan läggas på kartan.

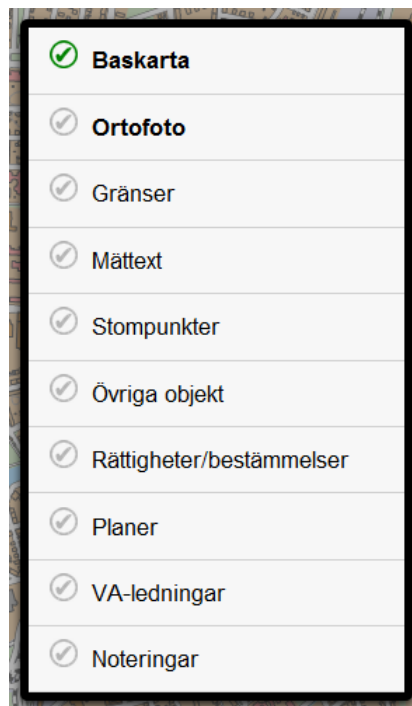


Figure 5: *Listan med valbara lager*

Baskartorna är i fetstil. Den karta och de lager som är aktiverade har en grön bock bredvid sig. Du trycker på det lager som du vill ska visas på kartan. Om du vill stänga av ett lager trycker du på det en gång till, så att bocken blir grå.

Under rubriken *Kartlager* finns vidare dokumentation av tillgängliga lager i appen.

3.1.1 Baskartor

Dessa lager är den baskarta som kommer att visas på skärmen. Bara ett av dessa lager kan vara aktivt och visas åt gången. Övriga lager kan sedan läggas på den baskarta som visas. Baslagren är:

- *Baskarta*: Baskarta med de väsentligaste lagren inlagda. De hämtas direkt från en TileCache, vilket innebär att laddning av kartan går snabbt och uppdateras ungefär en gång i veckan. Skulle du behöva det absolut senaste så får du be ansvarig om hjälp.
- *Ortofoto*: Baskarta med flygbilder.

3.1.2 Övriga lager

Dessa lager läggs ovanpå baskartan. Flera lager kan vara på samtidigt, men beroende på hur många som är på så kan prestandakänslan variera. Inga av dessa lager kan köras utan Internetuppkoppling. De övriga lagren är:

- Gränser
- Mätttext
- Stompunkter
- Övriga objekt
- Rättigheter/Bestämmelser
- Planer
- VA-ledningar
- Noteringar

3.2 Geolocate



Figure 6: *Geolocate-funktionen*

När du trycker på denna knapp så kommer din aktuella position att visas på kartan. Var medveten om att den position som visas är ungefärlig, en grå ring visar noggrannheten för tillfället. Du kommer alltså med största sannolikhet att befinna dig någonstans inom den grå ringen.

3.3 Sök



Figure 7: Sökfunktionen

När du trycker på denna knapp så presenteras en ruta på skärmen där du kan söka efter en fastighet, en adress, en stoppunkt eller en gränspunkt.

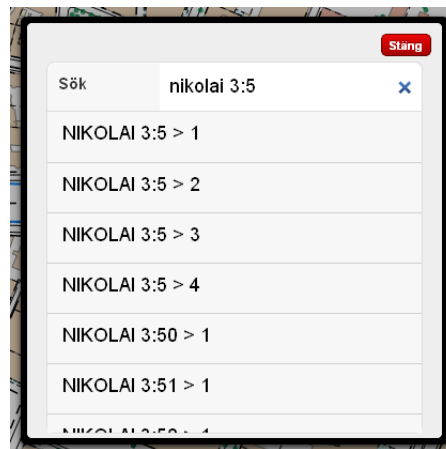


Figure 8: Resultatlista vid sökning

I sökfältet kan du skriva in namnet på det du söker, tryck sedan sök på tangentbordet. Om något påträffades visas det upp i listan. Om flera saker påträffades kan det vara nödvändigt att leta efter det i listan. Om inget påträffades lämnas listan tom. Om du trycker på något objekt i listan så kommer du att förflyttas till objektets position i kartan.

3.4 Export



Figure 9: *Exportfunktionen*

Efter att du har ritat färdigt föremål så ges en möjlighet att exportera det du har ritat. De objekt som du har ritat och exporterat kan sedan läggas in som ett konstruktionslager i GeoBas med hjälp av programmet impGeometrier när du är tillbaka på kontoret.

När du trycker på denna knapp så presenteras en ruta på skärmen där du ombedes skriva in ditt GeoBas-ID. Skriv in det och tryck sedan på OK. En ny flik i webbläsaren öppnas upp och på sidan visas det om exporteringen gått bra eller inte lyckades.

3.5 Lagring av kartvy



Figure 10: *Lagring av kartvy*

När du trycker på denna knapp så dyker en förfrågan upp på skärmen som frågar om du vill lagra kartområdet. Accepteras detta så kommer den aktuella kartvyn och dess underliggande zoomnivåer att sparas ('cachas') undan på den mobila enheten. Detta innebär att du kan få tillgång till denna vy och underliggande zoomnivåer ute i fält utan en aktiv Internetuppkoppling. Allt som inte är undansparat kommer att presenteras som grå rutor på kartan. Under tiden kartan lagras kan inget annat göras på kartan.

OBS: Endast baskartor går att lagra. Övriga lager går inte att spara.

3.6 Identifiera



Figure 11: Aktivera identifierafunktionen

När du trycker på denna knapp aktiveras identifieringstjänsten. Ikonen ändras då till ett kryss. För att sedan använda tjänsten trycker du i kartan på den detalj som ska identifieras. Var noga med att träffa rätt på de små objekten. För att tjänsten för ett objekt i ett visst kartlager ska fungera måste aktuellt lager vara aktiverat i lagerlistan. T ex, för att en gränspunkts attribut ska kunna identifieras måste lagret Gränser vara valt i lagerlistan.



Figure 12: Aktivera identifierafunktionen

För att stänga av identifierafunktionen så trycker du på krysset eller någon annan funktion.

Under fliken *Kartlager* finns vidare dokumentation av tillgängliga lager i appen och vilka av dem som kan identifieras med denna tjänst.

3.7 Markera



Figure 13: Markeringsfunktionen

När du trycker på denna knapp och sedan trycker på något objekt som du har ritat på kartan så dyker ett popup-fönster upp med lite information om det objekt du har ritat. Lite beroende på vilken sorts objekt du har tryckt på så visas olika information. Dels visas av vilken typ (punkt, linje eller polygon) som objektet är, samt om någon notering gjorts för det objektet. För punkter visas även deras koordinater, för linjer deras längd och för polygoner visas deras area. Det markerade objektet blir gult när det markeras.

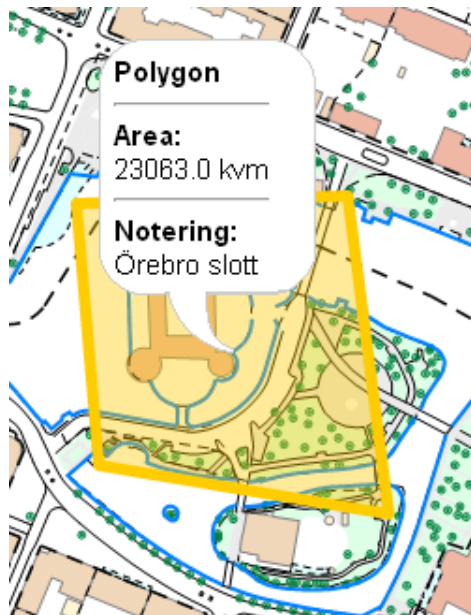


Figure 14: Markerad polygon och dess popup-ruta med markering

3.8 Notera



Figure 15: Noteringsfunktionen

Efter att du har ritat ett objekt så kan det vara till fördel att skriva en notering om objektet. Om du trycker på denna knapp så dyker en ruta upp på skärmen som ber dig att skriva en notering för ditt senast ritade objekt. Om du vill skriva en notering för ett objekt som inte är det som senast ritades så måste du först trycka på Markera-knappen

och sedan markera objektet som du önskar skriva notering för, och sedan trycka på Notera-knappen.

Tryck på fältet för att få upp tangentbordet och skriv sedan din notering. Efter att du har tryckt på OK så kommer noteringen att dyka upp under Notering i popuprutan som tillhör objektet. Efter att noteringen genomförts aktiveras Markeringsfunktionen automatiskt igen så att det blir lätt att markera nya objekt för ytterligare noteringar. Dessa noteringar kommer vid exportering att exporteras.

3.9 Hjälp



Figure 16: *Hjälpknappen öppnar manualen*

Om du trycker på den här knappen så öppnas en ny flik i webbläsaren upp, med manualen till iSurvey.

3.10 Statusfält



Figure 17: *Statusfältet visar aktuell information, t ex aktiverad funktion*

När en viss åtgärd utförs eller en viss funktion är under användning så kommer den aktuella åtgärden/funktionen att visas i detta fält i text.

3.11 Rita punkt



Figure 18: *Rita punkt*

Trycker du på denna knapp kan du rita punkter på kartan. Punkten ritas då på kartan på det ställe där du trycker.

3.12 Rita linje



Figure 19: *Rita punkt*

Trycker du på denna knapp kan du rita linjer på kartan. Brytpunkter på linjer ritas på kartan på det ställe där du trycker. Avsluta ritandet med ett dubbelklick på skärmen.

3.13 Rita polygon



Figure 20: *Rita punkt*

Trycker du på denna knapp kan du rita polygoner på kartan. Brytpunkter på polygoner ritas på kartan på det ställe där du trycker. Avsluta ritandet med ett dubbelklick på skärmen där den sista önskade punkten ska placeras. En linje mellan den första och sista punkten sluter då polygonen automatiskt.

3.14 Färg



Figure 21: *Byt objektfärg*

Du har möjlighet att välja färg på de objekt som du ritat. Ursprungligen är färgen inställd på blått, men då du trycker på den här knappen så presenteras en lista på skärmen med fyra olika valmöjligheter till färger.

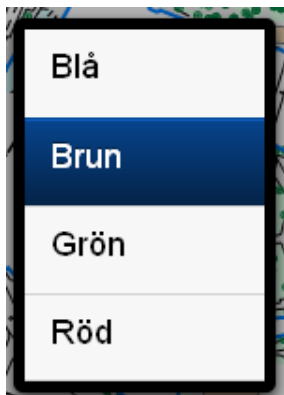


Figure 22: *Listan med valbara färger*

Färgerna som du kan välja mellan är blått, brunt, grönt eller rött. När du väljer en viss färg kommer alla objekt att ritas med den färgen. Knappen ändrar text beroende på vilken färg som valts.

3.15 Ta bort objekt



Figure 23: Ta bort ritat objekt

Trycker du på denna knapp och sedan trycker på ett objekt som du har ritat så tas det objektet bort från kartan. OBS: Du får ingen varning innan du tar bort ett objekt, så var försiktig så att du inte tar bort något objekt av misstag. Det går inte att ångra!

3.16 Ändra objekt

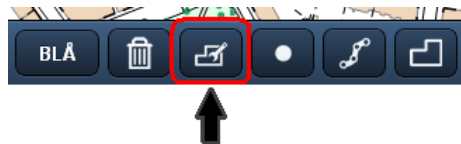


Figure 24: Ändra ritat objekt

Trycker du på denna knapp och sedan trycker på ett objekt som du har ritat så får du möjlighet att ändra på objektet. Nedan beskrivs på vilka sätt du kan ändra på olika typer av objekt.

3.16.1 Ändra punkt

Om du väljer att ändra en punkt innebär detta att du kan flytta på den till en ny position genom att trycka och hålla ned på punkten och sedan dra ditt finger på skärmen.

3.16.2 Ändra linje



Figure 25: *Linje under modifiering*

Om du väljer att ändra en linje så innebär detta att du kan:

- Flytta på den genom att på liknande sätt för punkt trycka på den punkt som dyker upp när du markerar objektet och flytta på det, samt
- Ändra dess utseende genom att trycka på de s.k. brytpunkterna och sedan dra dessa genom att trycka och hålla på dem och sedan dra dem med fingret.

3.16.3 Ändra polygon



Figure 26: *Polygon under modifiering*

Om du väljer att ändra en polygon så innebär detta att du kan:

- Flytta på den genom att på liknande sätt för punkt trycka på den punkt som dyker upp när du markerar objektet och flytta på det, samt
- Ändra dess utseende genom att trycka på de s.k. brytpunkterna och sedan dra dessa genom att trycka och hålla på dem och sedan dra dem med fingret.

4 Kartlager

4.1 Baskarta

Baskarta med de väsentligaste lagren inlagda. Hämtas direkt från en TileCache, vilket innebär att laddning av kartan går snabbt. Uppdateras ungefär en gång i veckan. Skulle du behöva det absolut senaste be ansvarig om hjälp. Endast denna karta går att lagra och köras utan Internetuppkoppling!

I detta lager ingår:

- Markanvändning
- Vägar: Identifierbara
- Järnvägar: Identifierbara
- Byggnader: Identifierbara
- Byggnadsdetaljer: Identifierbara
- Vägkanter: Identifierbara
- Strandlinje: Identifierbara
- Traktgräns: Identifierbara
- Adressnummer
- Gatunamn
- Traktnamn
- Träd: Identifierbara
- Fastighetsgräns: Identifierbara.

4.2 Ortofoto

Baskarta med flygbilder. Följande lager är identifierbara:

- Byggnader
- Byggnadsdetaljer
- Fastigheter.

4.3 Gränser

I detta lager ingår:

- Föreslagna gränser
- Gränspunkter: Identifierbara
- Gränspunktsnummer.

4.4 Mättext

I detta lager ingår:

- Mättext.

4.5 Stompunkter

I detta lager ingår:

- Stompunkter: Identifierbara
- Höjdfix: Identifierbara.

4.6 Övriga objekt

I detta lager ingår:

- Detaljmätta objekt: Identifierbara.

4.7 Rättigheter/bestämmelser

I detta lager ingår:

- Rättigheter: Identifierbara
- Bestämmelser: Identifierbara.

4.8 Planer

I detta lager ingår:

- Planer: Identifierbara.

4.9 VA-ledningar

I detta lager ingår:

- Vattenledningar: Identifierbara
- Dagvattenledningar: Identifierbara
- Spillvattenledningar: Identifierbara

4.10 Noteringar

Stänger av och sätter på de noteringar som gjorts för objekt i kartan. Sätts lagret på så visas alla noteringar som gjorts.

Institutionen av naturgeografi och ekosystemvetenskap, Lunds Universitet.

Student examensarbete (Seminarieuppsatser) i geografisk informationsteknik. Uppsatserna finns tillgängliga på institutionens geobibliotek, Sölvegatan 12, 223 62 LUND. Serien startade 2010. Hela listan och själva uppsatserna är även tillgängliga på LUP student papers och via Geobiblioteket (www.geobib.lu.se)

Serie examensarbete i geografisk informationsteknik

- 1 *Patrik Carlsson och Ulrik Nilsson* (2010) Tredimensionella GIS vid fastighetsförvaltning.
- 2 *Karin Ekman och Anna Felleson* (2010) Att välja grundläggande karttjänst – Utveckling av jämförelsemodell och testverktyg för utvärdering
- 3 *Jakob Mattsson* (2011) Synkronisering av vägdatabaser med KML och GeoRSS - En fallstudie i Trafikverkets verksamhet
- 4 *Patrik Andersson and Anders Jürisoo* (2011) Effective use of open source GIS in rural planning in South Africa
- 5 *Nariman Emamian och Martin Fredriksson* (2012) Visualisering av bygglovsärenden med hjälp av Open Source-verktyg - En undersökning kring hur man kan effektivisera ärendehantering med hjälp av en webbapplikation
- 6 *Gustav Ekstedt and Torkel Endoff* (2012) Design and Development of a Mobile GIS Application for Municipal Field Work
- 7 *Karl Söderberg* (2012) Smartphones and 3D Augmented Reality for disaster management - A study of smartphones ability to visualise 3D objects in augmented reality to aid emergency workers in disaster management
- 8 *Viktoria Strömberg* (2012) Volymberäkning i samhällsbyggnadsprojekt
- 9 *Daniel Persson* (2013) Lagring och webbaserad visualisering av 3D-stadsmodeller - En pilotstudie i Kristianstad kommun
- 10 *Danebjer Lisette och Nyberg Magdalena* (2013) Utbyte av geodata - studie av leveransstrukturer enligt Sveriges kommuner och landstings objekttypskatalog
- 11 *Alexander Quist* (2013) Undersökning och utveckling av ett mobilt GIS-system för kommunal verksamhet