CHALMERS

Traffic information on mobile platforms
- An evaluation using Android and push technology

Master of Science Thesis in Software Engineering and Technology

MAGNUS JONSSON
MATTIAS SVENSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, 2010

Traffic information on mobile platforms
An evaluation using Android and push technology

Magnus Jonsson
Mattias Svensson

Examiner: Per Zaring

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden

Abstract

This report is the result of a master thesis at Chalmers University of Technology. The purpose was to design and implement a traffic information system with focus on mobile platforms. The system interprets DATEX II data from the Swedish road administration (Trafikverket) and parses the data to a spatial aware database. Each system user can create a number of routes to travel and the parsed DATEX II data is pushed to the user's smart phone. The mobile phone application described in this thesis has been implemented in Android. The suggested system can be used in several other purposes thanks to its push technology and geographical aware database. Apart from the system this master thesis also researches the accuracy of one of the accelerometers commonly found in smart phones today as well as gives suggestions for further work and suggestions for future applications.

Table of contents

Preface/Acknowledgments

We would like to thank Crepido Systems and especially Michael Walenius for the opportunity to do our thesis at Crepido's Gothenburg office and for the support he has given us. We would also like to thank our supervisor at Chalmers, Per Zaring and all the persons that have helped us by setting aside time for interviews.

1. Introduction

1.1 Preamble

Daily commuting in a larger city is stressful, long queues and traffic jams are something that commuters have to deal with on a daily basis. Even the smallest accident can cause major traffic delays during heavy traffic. Two common ways for commuters to receive up-to-date information about road conditions is to listen to the radio or use a in-car-GPS that has a live Traffic Message Channel (TMC) feed. The problem is that none of these solutions are sufficient. Radio, for instance, chooses what information they want to broadcast to their listeners and this can be dictated by a number of factors such as time available for traffic reports, what will keep their listeners tuned in and many other things. This means that not all traffic information will be presented on the radio and road-users might miss information that could have made their journey easier and decreased travel time. The other example presented, in-car-GPS, will give the user all the traffic information that it receives, but the system used to deliver this information, Radio Data System-Traffic Message Channel (RDS-TMC) , does not cover all roads, at least not here in Sweden (Olsson, 2010). This means that the user again is not presented with all available information. Another issue with the in-car-GPS is that not everyone owns a in-car-GPS and even if one owns a in-car-GPS it is seldom used for daily commuting.

A system with more information sources and better ways to get this information to the road-users is desirable. The system should be able to gather information from several different sources and present this information to the user in a way that optimizes commuting. Mobile phones with a GPS, a large screen and constant internet connection get more common each day. Instead of using other devices one can use the mobile phones to get traffic information to the user. Apart from helping the road-user to commute there can be several other ways that a mobile phone, with a GPS, can assist road-users. One example of this could be a crash-detection application that uses the smart phones accelerometer to detect a car-crash and report its location. The possibilities with the accelerometer, in mobile phones, is something that will be further looked into.

1.2 Problem

As more and more people get smart phones with built in GPS and fast internet connection the need for a separate GPS device for navigation decreases. The smart phones are not able to receive traffic information through RDS-TMC, they instead need another way to receive this information. Since a smart phone allows for two-way communication the traffic information can be more targeted and a system for this is what we try to achieve with this master thesis.

1.3 Purpose

The idea behind this master thesis comes from Michael Walenius at Crepido Systems who had an idea on how to make traffic information more easily available for road users. Thus the aim of this master thesis will be to research a better way to handle and distribute traffic information to road-users and design a system that will satisfy this need. Michael’s idea concerned a lot of technologies but it was soon decided that the focus would be to come up with a general design for the system and how it should handle traffic information and make this available to users. A proof-of-concept version of an application for the Android platform will be developed to show how the information can be presented. Other necessary parts of the

system will also be implemented so that a functional proof-of-concept version of the system can be demonstrated.

This report will describe how one could work with the information from DATEX II and also show some of the possibilities with the Android platform, specifically how to push information to the user as it becomes available. Although the authors will design a traffic information system, which is quite a specific implementation, there are possibilities to generalize the system to handle other location specific information. The technique used for pushing and receiving messages could be used to allow the user to report his position at specific intervals and using this location push out relevant information. This information could for instance be location based advertisements such as restaurants, amusement parks, houses for sale etc. Other applications for the system will be discussed in section 8.3.

1.4 Limitations

The idea presented by Michael Walenius (see appendix C) covered many areas and is possible to implement on a large number of platforms, especially the mobile application. However this thesis will present a proof-of-concept system where the priority will be to get a functional system up and running instead of evaluating and researching which technology would fit the system best. Therefore this thesis will not in detail motivate why certain kind of techniques were chosen, this will be a proof-of-concept system and the time will be spent on the system idea in general. More research on specific technical solutions and security should be conducted if a live version of the software was to be implemented. To be able to go live with the system, all licenses of the used components also need to be checked, and some components might need to be substituted for others.

1.5 Audience

The authors of this report assume that the reader has knowledge about software development, databases and a general technical knowledge. The report will use many technical terms that will not be explained in detail.

2. Technical and theoretical background

Commuting in a larger city can be described in many different ways. Fun, efficient and peaceful are usually not words used to describe the commuting experience. With the arrival of technologies such as GPS and TMC several attempts has been made to make the life of commuters easier. By providing up-to-date information about the traffic conditions, better decisions can be taken by commuters and road-users in general.

2.1 Road-user behavior and the traffic information system

One important issue regarding traffic information systems is how the users respond to the information, if they respond at all. The fact that road-users react to traffic information has been established in several studies. What is more interesting is to know what kind of information they react to and what factors are important. Calculating the fastest route by simply calculating the fastest path is not sufficient (Abdel-aty et al., 1997). Road-users choice of route is effected by several other factors other than which way will be (in theory) fastest. In the study "Using stated preference data for studying the effect of advanced traffic information on drivers' route choice" (Abdel-aty et al., 1997) the effect of traffic information was investigated. The results showed both that road-users react to traffic information, and that travel time is not the dominant factor for choosing a route. The reliability of the information that was given to the road-users also played a significant role. Another study conducted at the University of Texas, Austin (Mahmassani & Liu, 1999) investigated how road-users decisions are affected by up-to-date information. The study focused on how day-to-day commuters' departure time and road decisions were affected by various types of information. One of their conclusions was that the reliability of the real-time information is very important. If the calculated arrival times given to the road-users were low more road-users changed their routes. This is important to think about when recommending new routes and calculating estimates, if the estimates are low the road-user will more often switch route. This can lead to an effect where a large number of road-users change their route because of low arrival estimates resulting in delays on that new route instead. So providing inaccurate traffic information can be worse than not providing any traffic information at all (Arnott et al., 1991)

Another important issue regarding a traffic information system is what kind of information that should be fed to the road-users. Too much information can discourage the user of the system and too little information can give an unreliable impression. The project team believes that it is important that the user can choose which kind of information he wants to receive and also in which manner to receive it. If the road-user is in a driving situation, clear, structured and precise information is needed. However, if the user would like to check road conditions before departure a larger number of different data sources could be presented.

Redelmeier, Tibshirani (Redelmeier & Tibshirani, 1997) and Bellavance (Bellavance, 2005) made studies on the association between phone calls and motor vehicle collisions. They found that, though there is a statistical relationship between talking on the phone and accidents, no causal relationship can be found. This means that we cannot say that talking on the phone causes accidents but there is at least a heightened risk when doing so. It does not matter if you use hand-held or a hands-free device (Sugano, 2005), it is the fact that you are focusing on the conversation that causes the distraction, not the fact that your hands are occupied. Since a lot of countries (Wikipedia, 2010) have adopted laws that prevent the use of mobile phones while driving, this needs to be taken into account when developing a traffic information system.

Our traffic information system is going to utilize a mobile application for presenting users with new traffic information and therefore needs to be carefully designed not to distract the user from driving and end up causing more accidents.

2.2 TMC and DATEX II

2.2.1 Background

TMC is a technology for delivering traffic information to road-users. TMC broadcasts live traffic information via the FM RDS. The TMC system is used in Europe and several other countries worldwide, the TMC feeds are usually controlled by each country's national road administration. This means that Trafikverket (formerly known as Vägverket) is responsible for the basic TMC feed in Sweden. TMC sends data messages which are then decoded by a TMC compatible car radio or navigation system. This means that the road-user can get up-to-date information about accidents, road work, traffic jams etc.

The TMC message broadcasting works like this (tmcforum.com, 2004);

- A traffic event (such as road work, traffic jam, accident etc.) occurs.
- The event is reported.
- The traffic information central logs the traffic event and creates a TMC event.
- The TMC event is broadcasted so that cars running GPS navigation with TMC can take another route.

2.2.2 TMC

The three most common systems to collect real-time traffic information is systems based on observation (for instance by helicopter), systems based on fixed sensors and systems based on GPS (Chouayakh, 2007). Sofiane Chouayakh has in her thesis investigated a fourth option to get traffic information, using mobile phones (Chouayakh, 2007). Mobile phones continuously send an Network Measurement Report (NMR) message to the mobile network. By using this message and split the mobile coverage into cells and mapping this against the road network an average speed can be calculated. However, the study showed some problems with accuracy and pointed out that the road-network in a city is too complex to be able to use the studied model.

The TMC feed can be provided by different companies, GPS manufacturers has contracts with different TMC providers to provide their GPSs' with a TMC feed. The different companies try to enhance the TMC feed to be able to deliver as much information as possible. These enhanced feeds can have the public information from Trafikverket as a base and then add more information from other sources. Destia Traffic is one example of such a company. They offer different kind of packages where the basic package includes only TMC data and where they also collect more advanced information that the purchaser can access. The collection of extra data is possible because of contracts with registered road agents, taxi companies etc. Destia Traffic collects all extra data in their own traffic central and the idea is that users then can subscribe to these information feeds.

The TMC system used today has some flaws. The TMC messages are sent without any form of encryption or authentication. This is quite interesting since users blindly trust the information and the fact that the information can be used to warn and redirect traffic in case of closed roads. TMC can also warn the road-user in case of airplane crashes or terrorist threats. A successful hack of the TMC could create chaos

although the people responsible for TMC claim that this is almost impossible it is still worth reflecting upon (Barisani & Bianco, 2007). The TMC system also suffers from the inaccuracy of the GPS system. The European TMC system relies on traffic points when locating events to get away from the GPS inaccuracy problem. Each country is given a number of traffic points and this means that the location where the event took place might not have a traffic point or the nearest point might be some distance from the actual event. This results in inaccurate positioning of the event and therefore affects the use of TMC in a GPS-device. During the interview with Lennart Olsson from Trafikverket (Olsson, 2010), the problems and future of TMC was discussed. The future European Galileo navigational system and a planned new protocol for TMC broadcasts will hopefully solve many of the problems with TMC today and we believe that this will make the TMC system more accurate and usable.

2.2.3 The DATEX II data

DATEX II is a standard that was developed to exchange information between traffic management centers mainly in Europe where the European Commission has been the leading development supporter (DATEX II.eu, n.d.). The DATEX II data comes in the form of Extensible Markup Language (XML) and details everything that needs to be known about the traffic situation.

Trafikverket provides public road-data and this is the data that our system will use. There are a number of data feeds that Trafikverket provides;

- Accident service - The accident service gives up-to-date information about accidents in Sweden. It gives information about the accident, road closures etc.
- Ferry service - The ferry service contains information about the different ferry connections in Sweden and their status.
- Rest area service - This service contains very detailed information about the rest areas in Sweden.
- Road condition overview - The road condition overview provides a by county description of road conditions. This information contains roads closed, weather related road conditions etc.
- Road condition service - The road condition service contains a large list of the roads in Sweden and their road conditions.
- Road work service - This service contains information about current roadwork.
- Traffic message service - Contains traffic messages (not accidents) such as queue information for Gothenburg.

Along with the different feeds that a user can access to retrieve information from Trafikverket it is also possible to request a push service from Trafikverket. This means that the user of the service sets up a web server with a web service that corresponds to Trafikverket's specifications and they will then push information to the server. The project team will primarily use the push technique. With every DATEX II data push from Trafikverket there can be one or more situations and for every situation there can be one or more situation records. These situation records can be of different types, for example an accident or an abnormal traffic. These records then contain detailed information about every situation, for example its position, severity and what it affects. It is these records that we parse for information and store in our database. From this data we will then be able to determine which route(s) these records are coupled with.

The project-team evaluated the different feeds and decided to temporarily leave out the road work service, road condition service and the ferry service. One problem with the different feeds is that there is a very large amount of data connected to each event (for instance an accident) and there is a very high number of

different possible data that can be sent depending on the type of traffic event. The project-team also discovered that the accuracy of the information varied. Figure 1 shows one of the inaccuracy problems with the data.

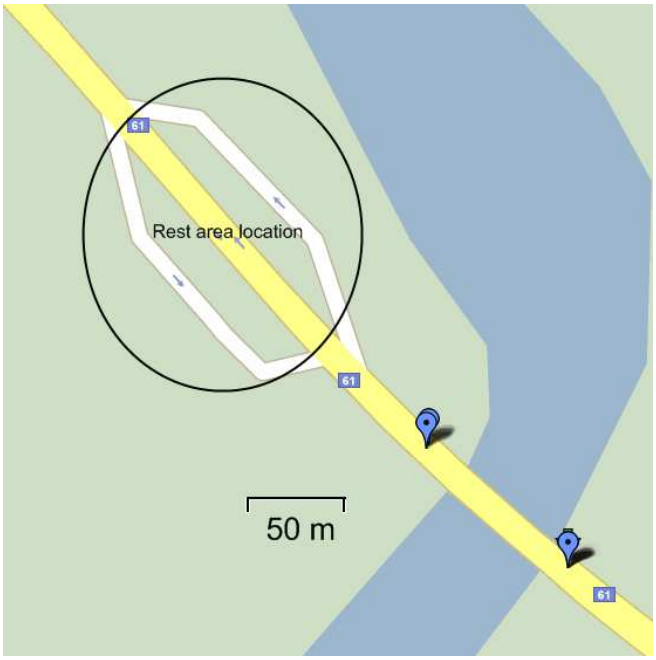


Figure 1. This figure describes the inaccuracy problem found in the rest area data from Trafikverket. The markers represent the position of the rest area according to Trafikverket's data and the circle the real location of the rest area. (Google Maps, 2010)

The picture shows a rest area (represented by the white side roads) that is accessible from both driving directions. The markers represent the GPS-coordinates for the rest area provided by the DATEX II data from Trafikverket. As the figure shows there is a problem with this data. The problem is that the coordinates are off by 50-200 meters. This is not a big problem when showing rest areas especially if the user is aware of the possible inaccuracy. However when it comes to accidents, accuracy is more important. If an accident occurs on a large freeway it is very important to know which side of the freeway the accident occurred. The accident could also have happened on a small road and it can then be very hard to determine in which direction the accident happened only by using GPS-coordinates, sometimes the coordinates can even point outside the road. As shown in this example, the latitude and longitude accuracy is not always 100% correct. Fortunately the case with the rest areas is a special case and the data for accidents is far more accurate than the rest area data.

The accident data manages to point on a road in most cases and has a much better accuracy then the rest area data. One explanation why the latitude and longitude is not always correct might be because Trafikverket primarily use another system for determining the location of an accident. The Swedish road grid is divided into several thousand different blocks; each block is represented by a traffic point. When an accident occurs the closest traffic point is selected and the offset is calculated. The different traffic points can then be looked up in a database that Trafikverket provides. We decided not to use the system with traffic points but instead only use the latitude and longitude provided for positioning. Our system will be designed to give the user information about accidents and since each traffic message has a very short description that tells on which road the accident occurred, different plain texts describing the accident,

latitude, longitude etc. The project team decided that this information would make the system accurate enough.

2.3 Maps and licensing

Before the design of the system started the project team looked into the problem concerning how events and objects should be displayed on a map. Google Maps felt like the obvious choice because of their high quality API and since Android is created by Google the two APIs should work fine together. We also thought about open alternatives such as OpenStreetMaps. OpenStreetMaps is open and free and its purpose is; "*OpenStreetMaps creates and provides free geographic data such as street maps to anyone who wants them."* (OpenStreetMap.org, 2010). Unfortunately the accuracy of OpenStreetMaps is not even close to the kind of map accuracy we would need. One example of this is if a user were to travel close to the small Swedish town Kil (Kil has about 8 000 inhabitants) they would see the map displayed in figure 2.

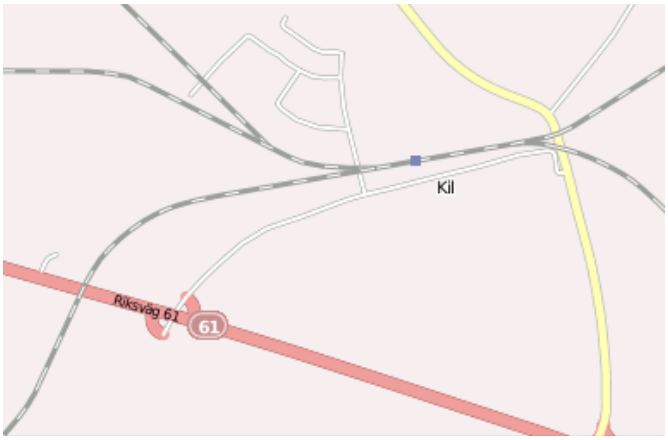


Figure 2. The town of Kil on OpenStreetMaps. (OpenStreetMap.org, 2010)

As one can see there is no trace of a town in figure 2 while the image from Google maps in figure 3 shows what the town of Kil really looks like.

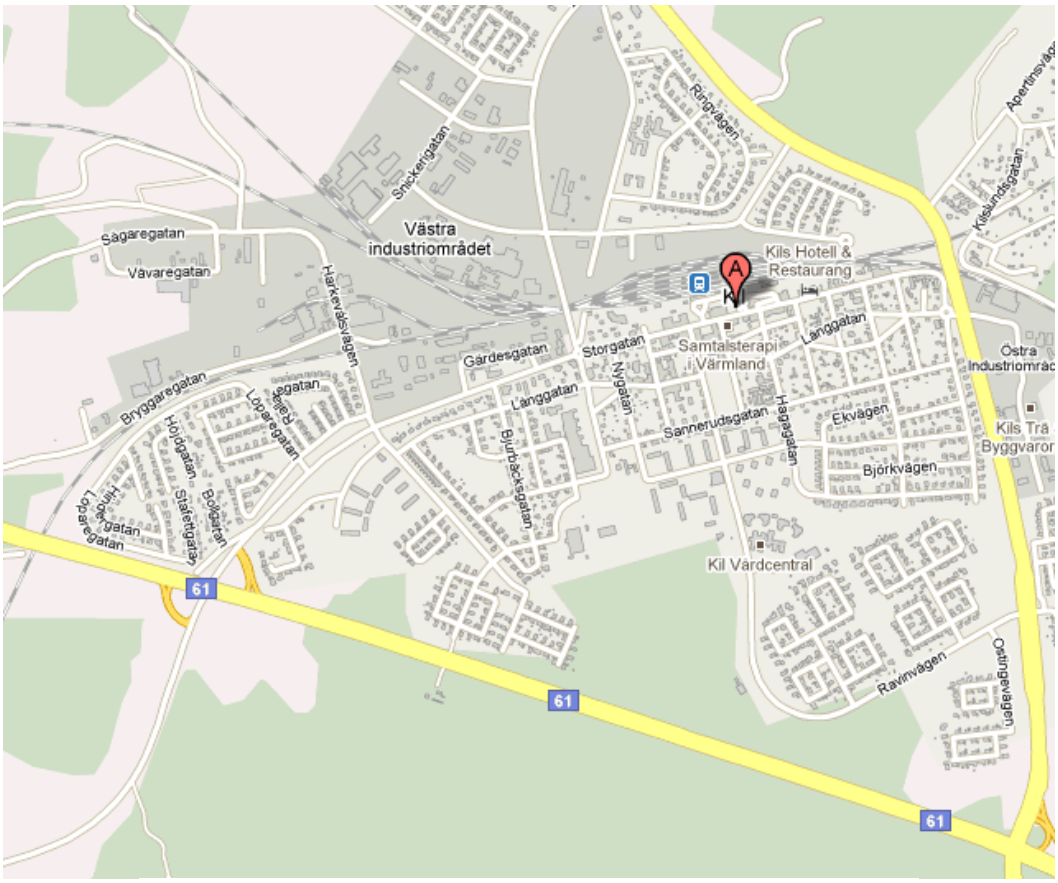


Figure 3. The town of Kil on Google Maps. (Google Maps, 2010)

This is just one example but there are several others where the maps in OpenStreetMaps are inaccurate. The project-team decided to use Google Maps. If the system will go live in the future it will be up to the owner of the system to decide what the business model should look like and decide what kind of map to use.

2.4 Using the accelerometer to save lives

2.4.1 Background

A study made in Finland (Sihvola et al., 2009) on the impact of having an automated emergency call system in cars found that such a system could probably (by 3.6%) help in reducing road fatalities. The system would work by dialing the emergency response center automatically when an accident is detected and transmit data about the accident and the location of it. This information would then help the emergency response center in sending out the right aid and to the right place. The study showed that the benefits would be the greatest when the driver is unable to call for help himself or when he cannot identify where he is. By having an automated system the time until an emergency call is made could also be decreased since it is not always possible for the driver to call himself and the accident might be on a road with sparse traffic where there are no eyewitnesses. It was also shown that though the system would help in preventing a large number of fatalities involving cars it could be even more effective when used with a motorcycle or a moped. Systems for alarming emergency services in case of a car crash can today be

found in some top-of-the-line car models from manufacturers such as BMW and Lexus. In these systems the car sends an emergency message to a call center and the call center personnel decides what kind of action to take. There is also a project called eCall that is a European commission project, there goal is to standardize the assistance systems in European cars. For these reasons it was investigated if some sort of crash detection system could be a part of our traffic information system.

The increased popularity of the cell phone in the past ten years has made it a lot easier to contact emergency services when an accident has occurred. However there are also problems related to using the mobile phone when calling in an accident, ambiguous or inaccurate location information and inundation of calls for the same accident has been a problem for the emergency responders (To & Choudhry, 2000). Today's smart phones are often equipped with a GPS and accelerometer and the use of these smart phones has increased over the past years. An application that makes use of the accelerometer and GPS to automatically alarm emergency services in case of an accident is now within reach. Such an application would rely heavily on the accuracy of the phone accelerometer and it is therefore important to learn more about the accelerometer and also about the forces involved in a car crash.

In a recent study (Ketabdar, 2009) a method for detecting physical shock with a mobile phone was investigated. The method used for interpreting the accelerometer (seen in figure 4) consisted of four different steps. By first processing the accelerometer data in a high pass filter the accelerometer noise was removed. The features needed from the accelerometer data was then selected and matched against a reference model. The reference models were built by data from normal use of a mobile phone and from different shock situations to be able to distinguish normal cases from shock cases. The final algorithm controls if the shock was followed by a period of low acceleration activity to sort out cases such as a phone being dropped and instantly picked up again. Even though these steps were taken to avoid false alarms the results showed that the accuracy rate was 91.1 %. This is, in our opinion, too low to be used for an application that automatically calls the emergency services.

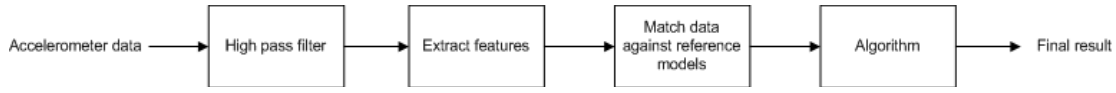


Figure 4. Method used for interpreting the accelerometer data in the study "Detecting Physical Shock by a Mobile Phone and its Applications in Security and Emergency" (Ketabdar, 2009)

There are other Android applications that claim to detect a car crash and they all have different ways of solving the accuracy and false positives problems. One project that tried to develop an application for detecting car crashes solved this by eliminating all shocks below 4G (To & Choudhry, 2000). Another way to solve this issue is to give the user a chance to cancel the alarm by for instance making the phone vibrate and make noises when it detects an accident. The user then has to push a series of buttons or perform some other task to cancel the alarm.

2.4.2 The HTC Magic accelerometer

The project team tested the accelerometer on a test phone, a HTC Magic (HTC Corporation, 2010), to get a better understanding of it. The accelerometer API has four different sensitivity settings, fastest, game, ui and normal. A test program was written that simply calculates the acceleration by comparing the current acceleration value with the previous one. The program ran five seconds on each sensitivity setting. The readings were collected by shaking the phone back and forth for the 20 seconds that the test program ran.

These tests were conducted to get a general idea of how the accelerometer worked and to learn more about its precision. The first runs of the test program showed that, although the shaking movement (the phone was shock back and forth at "normal" speed) was about the same, the acceleration varied from around 30 m/s^2 to 1000 m/s^2. To get a better understanding of these numbers Table 1 lists three different examples of G-forces. 1 G is equivalent to about 9.82 m/s^2 and it is therefore very doubtful that we, by shaking the phone, could have reached accelerations over 500 m/s^2.

Example	**G-force**
Standing on the earth	1G
High-g roller coaster	3.5–6.3 G
High speed car collision	40-50 G (vcu.edu, n.d.)

Table 1. Examples of the G-forces experienced in three different situations.

The first test results can be seen in figure 5. One can see that there were some peeks in acceleration, with values sometimes more than doubling the previous value. These peaks occurred the first second after the change in the sensitivity settings. During this second the sample rate was very uneven.

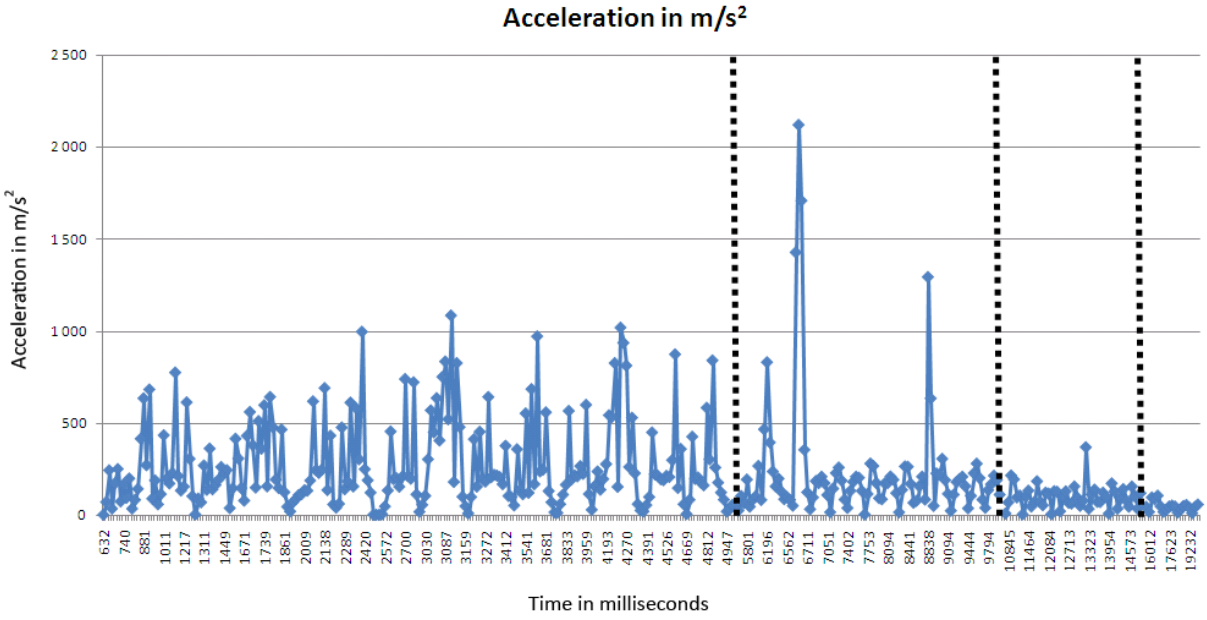


Figure 5. Diagram representing the first accelerometer test run. The dotted lines marks the time when the sensitivity settings were changed.

For the second test run the test program was adjusted. Instead of using the system time in milliseconds the time stamp that each movement had (in nanoseconds) was used. This gave a much better and even view of the sample rate. The first second after each frequency change was also ignored because of the bad data during that period. As can be seen in figure 6, the results were still pretty bad and the project team believes that further testing is required before considering using the accelerometer in an emergency notification system. Implementing filters and using some sort of test rig to calculate the accuracy of the

accelerometer could be a start but as seen in the report by Ketabdar (Ketabdar, 2009) it is hard to get a high certainty level. The project team believes that the certainty level needs to be considerable higher than 91%.

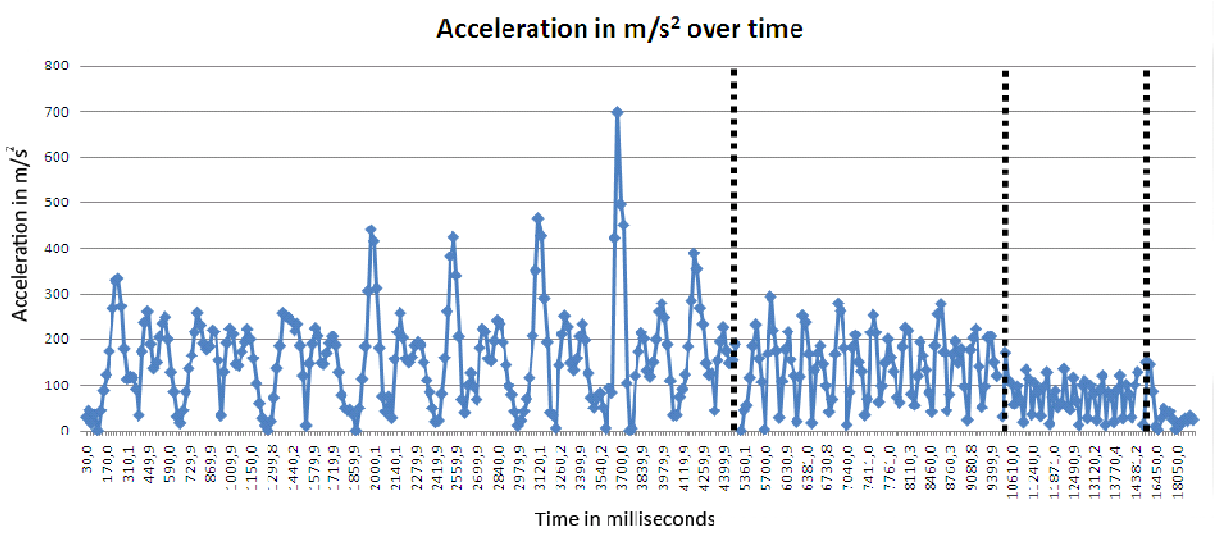


Figure 6. Diagram representing the second accelerometer test run. The dotted lines marks the time when the sensitivity settings were changed.

The test program was run several times. The sample rate was uneven but after the programs time tracking technique was adjusted, so that each accelerometer readings timestamp was used instead of the system time, more stable sample rates was received. The sample rates can be seen in table 2.

Sensor delay	Sample rate [ms]
Fastest	20
Game	50
UI	100
Normal	250

Table 2. List of the HTC Magic accelerometer sample rates on different sensor delay settings.

Our limited testing shows that both the sample rate and the readings are uneven. The best results came when the sensor sensitivity was set low. This is a problem since a car crash is over in approximately 70 ms (Colquhoun, 2008) and the sensitivity settings sample rate that gave the best results was 250 ms and 100 ms. If a car crash detection application was to be developed, further testing should be conducted to make sure that it really detects a car crash.

3. Case Study

When we finished the first system description we conducted a small survey to see if there were aspects of the system that should be changed and to get comments about the system idea. The survey (see appendix E) gives input about different aspects of the system and especially about the user interface.

3.1 Crepido Systems AB

Crepido Systems is an IT consultant company that develops and integrates data systems and mobile solutions designed to the customer needs. Crepido has offices in Växjö, Gothenburg, Skövde and Västervik with a total of 35 employees. Crepido's business idea is to combine business thinking with technical state of the art competence and the ability to choose the right technique for the client. This technique does not have to be the latest and most sophisticated but rather technique that has been proven reliable which results in a system that will be easy to maintain. As mentioned earlier Crepido mainly develops in three different areas, these "three legs" is the foundation that Crepido stands on. The areas are system integration, system development and mobility.

The idea for the master thesis comes from Michael Walenius at Crepido's Gothenburg office. Michael has a concept idea about computer software that handles traffic information. The concept is very broad and our objective is to research this concept to see what works and what does not work and in the end come up with a system design and implement parts of the system. The traffic information system is a system that helps road-users by making sure that they always have up-to-date information about the current traffic conditions along their route. The user gets information about the current situation from a central server to a mobile phone application on his mobile phone. Apart from the mobile application the system also consists of a web page where the users can change their settings, create routes etc.

4. Methodology

This project is of a more experimental nature and there are only vague guidelines and requirements for the product. The project team will explore several different ideas and will therefore use an evolutionary development process (described in figure 7). The development will be of a throw-away prototype nature which means that we start off with vague requirements and prototype our way to a finished application. This is especially good for this project since evolutionary software development gives the customer an early release of the software and gives fast feedback of any changes made to the software (McCormack, 2001). Since we have received a concept idea from Michael Walenius it will be good to be able to show the progress and system design on a regular basis.

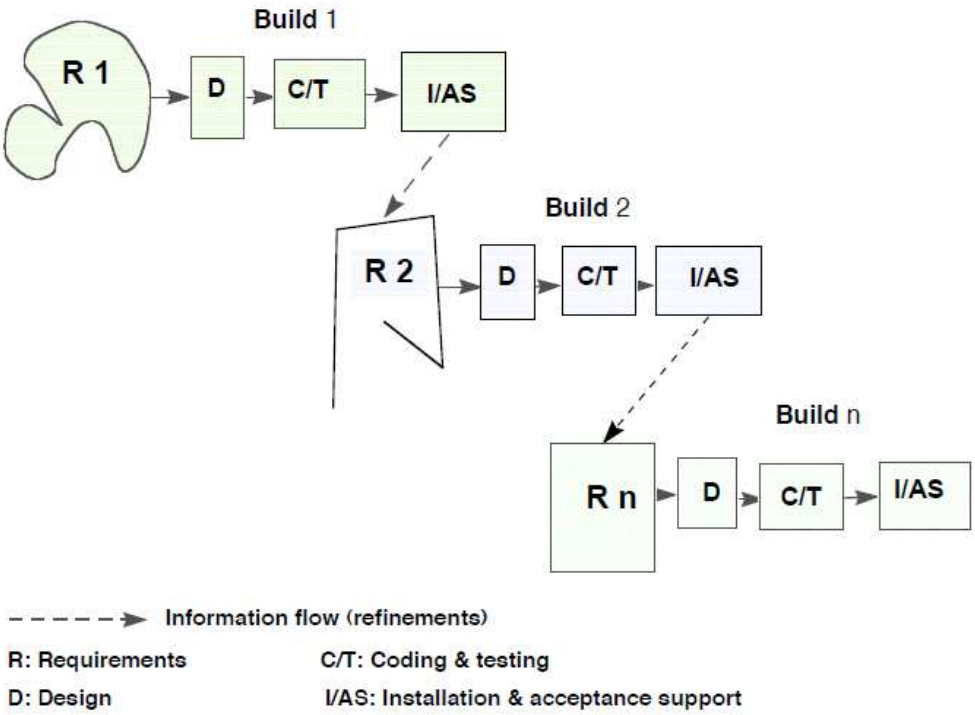


Figure 7. The figure shows the evolutionary process used in this project. The project goes from vague requirements in R1, better requirements in R2 and finally during the last iteration the requirements are complete. (ISO International Organization for Standardization, 2008)

The project team has also chosen to work according to an evolutionary process when it comes to time planning. Since the project starts with a concept there are several uncertainties in the project and when facing several uncertainties there is no need to try to formulate an exact time plan since it most certainly will change during the project (Maylor, 2005). A rough initial time plan was created and it was decided that the project would be planned in five-week intervals. This time management technique is called time boxing. Time boxing is typically used for software development projects and each time period in a time box is normally two to six weeks long. Our time boxing strategy will, as mentioned, be to divide the work into five week intervals. After each time box one week of documentation and evaluation of the five-week period will be conducted. When the past five-week period has been evaluated and documented a detailed plan for the next five weeks will be created. Because of this workflow there will be no specific documentation phase until the end of the project. The original time plan can be seen in appendix B.

4.1 Project phases

When the project is finished and the final iteration completed the project will have gone through the following phases.

- Prestudy.
- Concept scope analysis and breakdown.
- Research and limitations.
- System design.
- Implementation.
    - Iteration 1 - Android application.
    - Iteration 2 – Back end.
    - Iteration 3 - Connecting Android application with back end.
    - Iteration 4 - Refactoring.
    - Iteration 5 - Testing.
- Documentation.

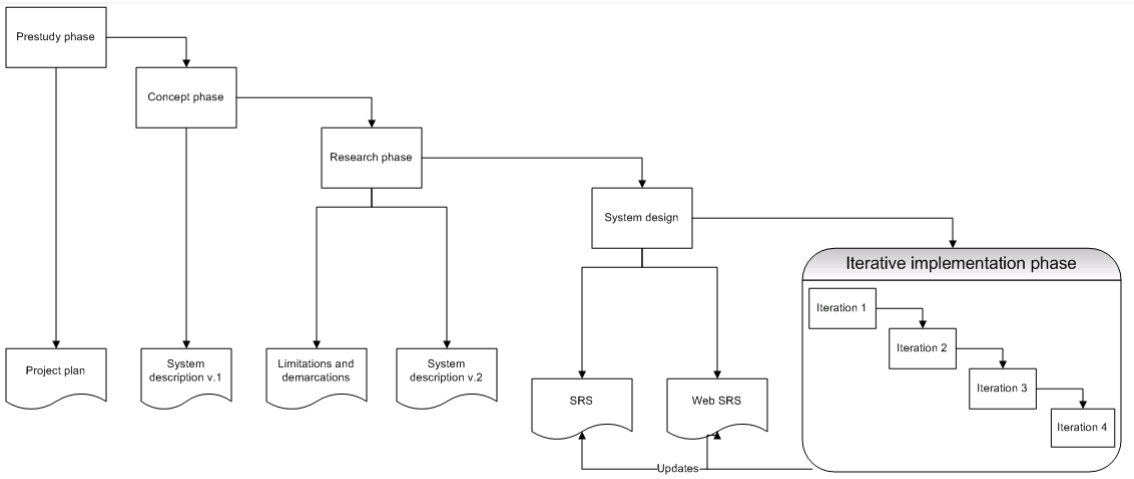


Figure 8. Project work flow

4.2 Documentation

Although documentation and planning had been conducted in roughly five week intervals the last part of the project was set aside for documentation. Since this is a master thesis it was not only the code and project that needed documentation. The final master thesis report needed a lot of work and this was therefore the main focus at the end of the project. However, the continuous work with the report went over expectations and this made it possible for a fifth implementation iteration.

4.3 Distribution of work

The master thesis work has been conducted at Crepido's Gothenburg office. This meant that the project team could work side by side during the entire project which lead to a natural division of the work load. The different tasks was divided equally time wise, however, Magnus took a bigger responsibility when it came to planning, setting up meetings and writing documentation while Mattias took a bigger responsibility when it came to the technical parts of the project.

5. Execution

5.1 Prestudy

The prestudy phase's first objective was to talk to Michael Walenius at Crepido to get a good overview of his concept idea. When the project team had a good view of the concept and how to carry out the work the project plan was created.

Deliverable: Project plan and time plan, see appendix A and appendix B.

5.2 Concept scope analysis and breakdown

The first step in the concept phase was the concept brainstorming. During the brainstorming the original concept idea (shown in figure 9) was evaluated.

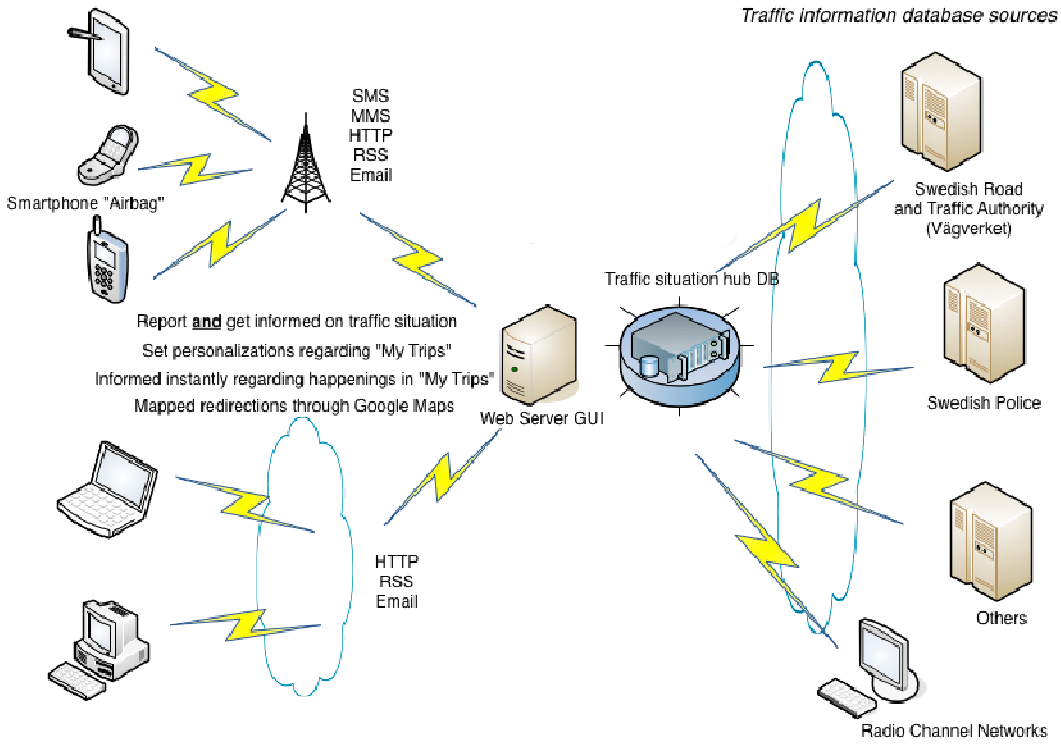


Figure 9. Original concept idea created by Michael Walenius.

The brainstorming and discussions concerning the concept idea resulted in a preliminary system description, which was discussed with Michael Walenius. This preliminary system description presented the major system concept ideas and gave a conceptual description of how the system could be designed and what it would contain. Finally a list of different subjects to research was created from the system description.

Deliverable: System description v.1, see appendix C.

5.3 Research and limitations

The research phase was dedicated to the gathering of relevant information and conducting interviews and surveys. The results from the research phase are described in further detail in the research part of this report (see section 2). The research concerned studies about different systems concerning traffic messages, such as GPS and TMC. The project-team also researched road-user behavior and how traffic messages affect road-users. At the end of the research and limitations phase enough information had been gathered to re-evaluate the preliminary system description, make changes and finalize it.

Deliverable: System description v.2 with limitations and demarcations and a survey result document, see appendix D and appendix E.

5.4 System design

During the system design phase the project team created the initial system requirement specification (SRS). During this work the project team decided to extract the web interface and Android requirement specifications to separate SRSs.

Deliverable: Web SRS and Android SRS, see appendix F and G.

6 Implementation

6.1 Iteration 1 – Android application

During the first iteration the project team first devoted some time to getting acquainted with the Android platform. The recommended developing platform for Android is Eclipse (developer.android.com, 2010) and since both members of the project-team is used to working with Eclipse it was decided to use Eclipse for the development.

During this iteration the graphical user interface (GUI) of the Android application emerged. The functional requirements were available but there was nothing which specified how the GUI should look. The goal for the GUI was that it should feel intuitive for the user of the application which could be many different types of people, from young to old, with varying degrees of technical experience.

The first attempt at a GUI incorporated both the traffic information part and the crash part of our initial idea. A design that was later discarded in favor of two separate applications. The final GUI instead consists of three different screens from which a user can reach all necessary functionality. For a full description of the GUI see appendix H.

The first screen of the application is a start screen where the user can change the settings of the application and select the desired route. This takes the user to a screen where he can see gathered information which can help him before he begins his travel. The third and last screen holds the information and functionality that the user can benefit from while he is driving.

6.1.1 Retrieving information

The Android application relies on a back end server, described in section 6.2, where it can retrieve the necessary information and present it to the user. From this server the application retrieves the user's routes, recent traffic messages and a road condition overview for his current location. The retrieved information comes in the form of XML. To receive live traffic information while driving the Android application keeps an open connection to an MQTT message broker which is used to push information from the back end to the mobile phone.

6.2 Iteration 2 – Back end

During the second iteration of the project the focus lied on handling the data from Trafikverket and creating a working back end for the system. A simple web interface to handle creation, editing and deletion of routes were also developed.

6.2.1 Working with the data from Trafikverket

Trafikverket has two options for getting hold of their data, either you pull it from their server or they push it to your server. We opted to begin with pulling data from their server since this allows us to more easily test our application as we can request data when we like instead of having to wait for them to push new data. In the end we implemented a web service for Trafikverket to push traffic information to. This allows the system to be fully automatic without us having to request new information on a regular basis.

From Trafikverket's wiki page we could download some example files and we also got Web Services Description Language (WSDL) files for both push and pull, along with XML Schemas, to use when requesting and receiving traffic information. We took the WSDL files and from these we generated Java classes that we could use to request information from Trafikverket.

What one gets as the result from a request is a data structure that holds all the information for one or more traffic events. We have created parsers for this data structure to retrieve the information that we need for our system, and store this in the database. These traffic events are then matched against routes that users are currently driving on and delivered to them almost instantaneous.

6.2.2 Web interface

As stated previously we developed a simple web interface where users can manage their routes. To be able to use Google Maps in a simple way using Java we used Google Web Tools (GWT) (code.google.com, 2010) to create the web interface. GWT allows writing code in Java and then having it compiled to JavaScript. GWT also optimizes the code that it generates and makes sure that it is compatible with the major browsers. The focus has not been on the design of the user interface for this proof-of-concept implementation; it is just a way of exposing the functionality of the system.

6.2.3 MQTT (MQ Telemetry Transport)

We use MQTT for pushing traffic information to the Android application.

"The MQTT protocol enables a publish/subscribe messaging model in an extremely lightweight way. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium." (mqtt.org, 2010).

This allows us to keep an open connection from the phone to the message broker in an easy way. The message broker is the application that routes all the messages to the correct recipients. The phone subscribes to a specific topic, based on the route selected, with the message broker. The server publishes traffic events to that topic and they are sent to the phone by the message broker. By using MQTT we can easily implement two-way communication if we allow the Android application to publish information to topics as well.

Using MQTT does come with some side effects. The broker is open to anybody as long as you have an MQTT client. This means that anybody can listen on the traffic messages sent to different routes. This might not be entirely bad since this allows users who do not only want to use their mobile phone for reading the traffic messages to create their own implementations. If one do not want to allow users to create their own implementations you would have to restrict access to the message broker through some form of authentication. A simpler approach might be to create a more complex topic name for each route. This would still allow users to quite easily figure out the topic name for their own routes but it would make it harder to figure out the topic names for other people's routes.

Another side effect that is more serious is the fact that the MQTT broker allows anybody to post their own traffic messages just by connecting to the message broker and sending correctly formatted XML for a traffic message to the correct topic. This could be prevented by only allowing specific IP-addresses to post

to the message broker. If the service that pushes the messages to the broker resides on the same computer as the broker you could restrict it to only allow the posting of traffic message from the local host. This is something that needs to be corrected if the system should go live.

6.2.4 Database for the back end

We tried a couple of different ideas for the database back end before we settled on using PostGIS (PostGIS, n.d.).

Our initial attempts

Our first attempt at a back end used SQLite for storing traffic information and the user's routes. Traffic information in our system has latitude and longitude, besides all the other information, for where the incident has occurred. The routes would be made up of a number of points that when connected with straight lines would make up the route. These points were generated by Google Maps when the user created his route.

Our initial idea for checking if a traffic event should be connected to a route (reported to the user of that route) was to take one point on the route and also the next one along and from those two points create a rectangle and if the traffic event was contained in that rectangle it would be connected to that route. This posed a number of problems. The most obvious being that it would be really inaccurate and it would probably also miss a few events that should be connected to a route.

Suppose that we have two points like in figure 10.

Figure 10. Two points along a route and the bounding box they create

As point A and B lies directly above one another we do not get a rectangle. Instead we get a line and for a traffic event to be connected to the route it needs to be exactly on that line. Since the location we get from Trafikverket is not always accurate we could end up not connecting a few events to routes that they should be connected to.

Again, suppose we have two points like in figure 11.

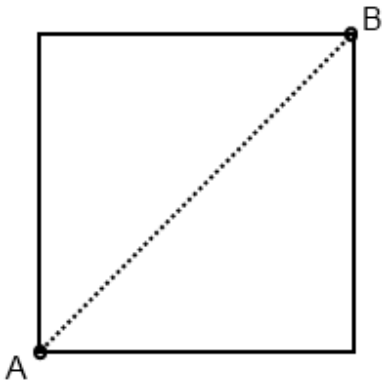


Figure 11. Two points along a route and the bounding box they create

Point A and B are far apart (many kilometers) and they lie so that they create a square. If we connect every traffic event that lies in this square to the route we would probably connect a few events that actually happened on another road, especially if it is in a city, since we would connect traffic events that could be a few kilometers away from the road. This first idea was quickly discarded since it would result in a really bad and frustrating system for the user.

Our next attempt was to instead of creating rectangles from the points draw lines between them and check the distance to that line, see figure 12. This would give us a good accuracy and minimize the number of false positives when connecting a traffic event to a route, if we selected a low enough distance from the line to use. This idea ended up being quite slow and while looking for a way to speed it up we came across PostGIS. PostGIS add support for spatial objects to the PostgreSQL object-relational database.

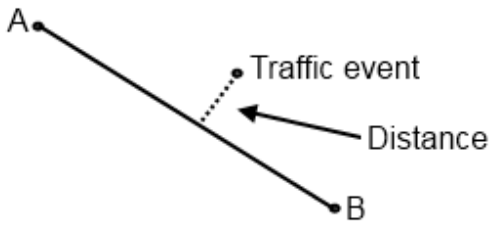


Figure 12 Two points along a route and an imaginary line between them used to calculate the distance to a traffic event.

The final database system

PostGIS seemed like a good fit for our purpose and this is the database back end that we decided to use. Using PostGIS meant we had to adapt our code to make it use the new geometry objects in the database. For our first attempt we stored a route as one long Linestring, one of the types supported by PostGIS. This however turned out to be very slow and the documentation for PostGIS told us why:

"Current PostgreSQL versions (including 8.0) suffer from a query optimizer weakness regarding TOAST tables. TOAST tables are a kind of "extension room" used to store large (in the sense of data size) values that do not fit into normal data pages (like long texts, images or complex geometries with lots of vertices), see http://www.postgresql.org/docs/current/interactive/storage-toast.html for more information).

The problem appears if you happen to have a table with rather large geometries, but not too much rows of them (like a table containing the boundaries of all European countries in high resolution). Then the table itself is small, but it uses lots of TOAST space. In our example case, the table itself had about 80 rows and used only 3 data pages, but the TOAST table used 8225 pages." (PostGIS Ch.6, n.d.)

It seemed that when we dealt with large geometries the queries would be slow and a route from the north of Sweden to the south of Sweden would consist of about ten thousand points. So we decided to rethink our approach again. Instead of storing one large Linestring for every route we split it up in many smaller ones by creating one Linestring between every point on the route and the next one. This resulted in a faster query and this is our final solution to this problem.

When using PostGIS one can ask queries like: "Give me all routes that lie within 10 meters of this point". This makes it easier for us to use and we can leave the optimization of algorithms to those that are more suitable and experienced in that kind of work. Since PostGIS is compliant with the Open Geospatial Consortium (OGC) Simple Features (opengeospatial.org, 2010) there are a number of libraries and software to work with this kind of data.

We use Hibernate (hibernate.org, n.d.) and Hibernate Spatial (hibernatespatial.org, 2010) to work with the data from the database. Hibernate enables object/relational mapping to allow us to easily store objects in a database. It is a well used framework and it has a lot of support. Hibernate alone cannot work with spatial data since it is an extension to the standard types defined for databases. This is however why we have Hibernate Spatial (hibernatespatial.org, 2010), this allows us to access spatial data and make queries related to it. Hibernate Spatial implements most functions of the OGC Simple Feature Specification.

6.3 Iteration 3 - Connecting Android application with back end

During this iteration our focus lie on connecting the two parts previously created; the back end with traffic information and the Android application for user interaction. As detailed previously we use a MQTT message broker to handle the communication between the server and the phone. The information flow between these components is designed in the following way;

When a new traffic event is received it is stored in the database to keep it available for later use. The location of the new event is then checked against the currently active routes (no need to check against all routes). When a route that the event should be connected to is found the message is published to the topic for that route and received by the Android application. The traffic event is sent as a XML document containing the necessary information for the user. When the message is received in the Android application the XML file is parsed and the traffic event is displayed to the user. The Web interface is connected with the database for the user to be able to manage routes. An overview of the system can be seen in figure 13 and a more detailed overview can be seen in figure 14.

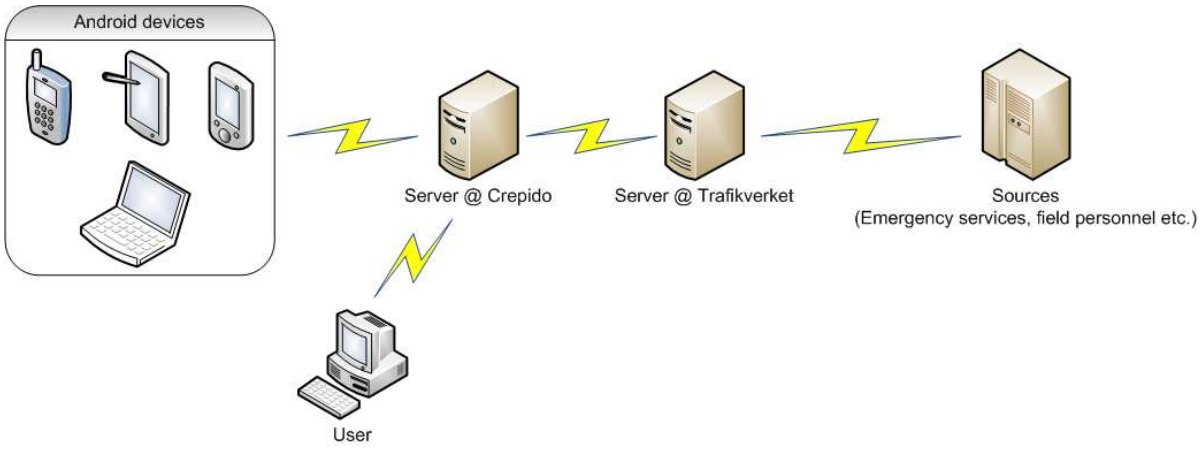


Figure 13. System overview

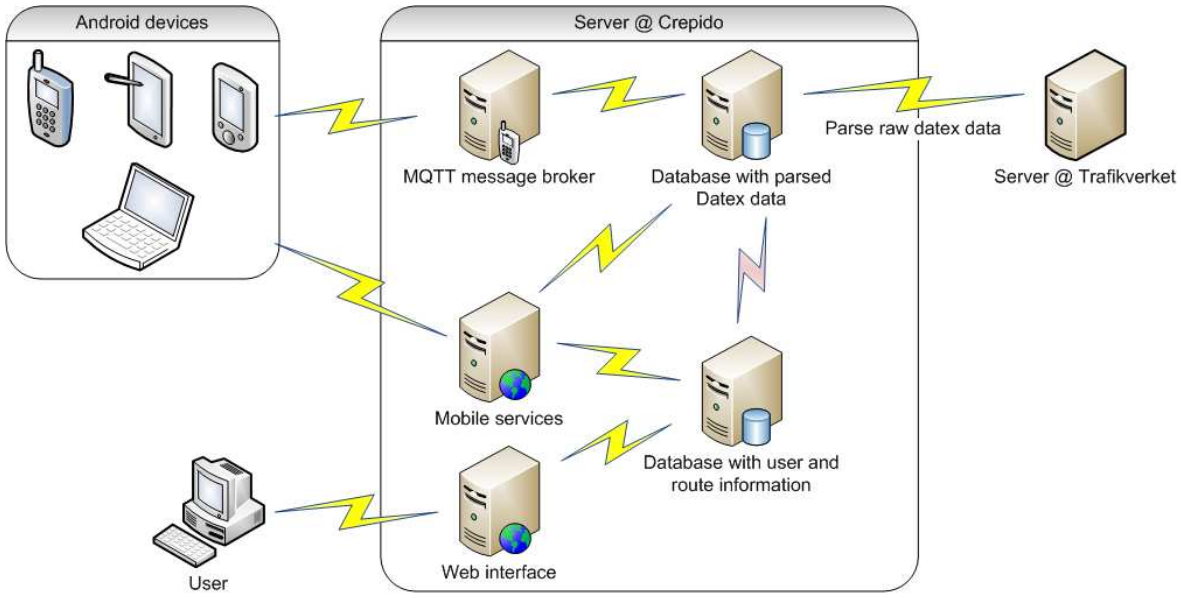


Figure 14. Detailed system overview.

6.4 Iteration 4 - Refactoring

Because of the evolutionary process and the fact that we had not coded Android before an iteration that focused mostly on refactoring was needed. During this iteration the code was cleaned up and the application was also more thoroughly tested and error handling was conducted. This was also when we first got the opportunity to test the application on a real device, an HTC Magic running Android 1.5.

6.5 Iteration 5 - Testing

Iteration five was not part of the original plan but as time became available it was decided that a fifth iteration was to be conducted. During the fifth iteration more testing and refactoring of the system was done and some parts of the system were rethought. Some changes in the MQTT topics used were made and some additional features were added. The additional features and applications that were designed and implemented were test applications to show what could be done with the parsed DATEX II data. One application had the sole purpose of displaying traffic queues around the Gothenburg area.

7. Result

7.1 Research conclusion

This report has shown how smart phones could be used for two-way communication in a traffic information system. The authors' research concluded that several studies have shown that road-users react to traffic information when choosing routes. However, simply calculating the fastest-path is not sufficient since the road-users decisions are effected by several other factors. The reliability of the information is important and it is very important that the system does not display false information or inaccurate calculations. The research also showed that when it comes to system design it is important to think about the consequences of using a mobile phone while driving. Studies have shown that it does not matter if you use a hand-held or a hands-free device, it is the fact that you are focusing on the conversation that causes the distraction, not the fact that your hands are occupied. In a traffic information system where a smart phone is used in the car by a road-user the system has to be designed to minimize user distraction and be as easy to use as possible.

This particular system uses the DATEX II data provided by the Swedish road administration, Trafikverket. The data is not always 100% accurate and does not have as many different information sources as some other providers of traffic data. However, the data from Trafikverket is available without any fees and Trafikverket is not a profit driven company or organization which is vital since this is an important service and the long term availability needs to be high.

The initial concept idea for the traffic information system included a crash detection application. There are several crash detection applications for Android on Android Market but the project team has failed to find any information about how the applications has been tested to make sure that they detected a car crash. From the tests conducted by us (see section 2.4.2) we find it hard to believe that the phone accelerometer is trustworthy enough for a car crash detection application.

7.2 Product description

The final system consists of a central part which will retrieve, handle and distribute traffic information. This central part will be the one that makes sure that the road-users are provided with current information about traffic conditions. Apart from the traffic information this part of the system will also be responsible for handling user information and settings, which can be configured through interfaces. For further information see section 6.2.

The system has a web interface that is used by the user to create routes that the user travels frequently or for some other reason wants stored in the system. These routes can then be selected in the mobile application when a road-user is about to begin his travel and he will be presented with current information on traffic events along his route as they are reported, for further information see appendix F.

The back end part of the system is centralized around the PostGIS database. It is this database that does all the heavy lifting when it comes to determining if a traffic event is connected to a route. We decided to go this way since it allowed us to rely on algorithms developed by people with more experience in the field. Our own first attempts did not match up against PostGIS, see section 6.2.4.

The mobile application part of the system has been implemented in Android. The decision was made since both authors were interested in Android development and none of us owned a Mac, which is needed for iPhone development (Android and iPhone were the two platforms discussed). Getting started with the development of Android applications was easy and the development environment and the Android API are easy to work with. The mobile application is not intended to do that much processing of traffic information (it does fetch the rest areas from a local database but that's it). The application is mostly intended as a front end to the traffic information system for presentation of data. All the logic of which traffic event is connected to which route is handled on the back end. For more information see section 6.2.4.

To facilitate the communication between the back end and the mobile application the system uses the MQTT protocol. The MQTT protocol enables a publish/subscribe messaging model and allows for easy and lightweight communication between the two parts of the system. For a more detailed description of the MQTT protocol refer to section 6.2.3.

A detailed overview of the system can be found in figure 14, in section 6.3.

8. Conclusion

8.1 Resume

We were given an idea about a traffic information system that would push only the relevant information (traffic events along a user's route) to a user's mobile phone. From this initial idea a general design was formulated addressing the needs and possibilities of the system. The idea was to create a proof-of-concept system to show that is was possible to take in information from different sources and push them to a mobile phone as soon as it is available.

The problem needed researching to see what effects traffic information reports could have on road-users. Studies have shown people take into account traffic information reports when planning their routes, if they feel that the source is reliable. Our research showed that traffic information when handled correct can be a powerful tool to help road-users. We have also shown the limitations of the accelerometer and we hope that further research around this subject will be conducted.

We have not developed a system ready for public use but we have shown that (and how) such a system could be developed. Many parts of the system developed during this master thesis can be directly used in a live version but some will need to be changed or reimplemented.

8.2 Discussion

Since we were inexperienced in both Android development and Web development we were bound to make some mistakes along the way. We started out trying to come up with a good design for the system but soon found out that it was somewhat difficult when you are inexperienced in what you are trying to do. What usually happened was that we followed our initial plan until we hit a problem and then started reading up on possible solutions. After choosing the solution that we thought would be the best one we started to implement that solution. Because of this we often came into a spiral where we hit a problem, researched, solved the problem, hit a problem, researched etc. Our focus was mostly on getting a functional system and even though we tried to have as clean code as possible we sometimes just fixed the problem as quick as we could which sometimes resulted in messy code or a suboptimal solution.

When we started out we wanted a nicely designed system and we tried to take it in that direction but in the end we wound up with some more or less coupled systems. Sometimes this was necessary, like the MQTT broker, but other times we probably could have integrated the different parts more closely. If we got the chance to start over we probably would have done some things differently. These problems are related to the fact that we wanted to implement as many parts of the system as possible to show how this concept could work.

The projects division of work (see 4.2.7) came natural and we did therefore not put much effort in planning the work division. Even though both project members put in the same amount of time the amount of work in the different areas varied. This came natural since we have different interests and skills. However, this also means that sometimes one project member took full responsibility of a certain part of the project and finished this while the other project member had little knowledge about how that part functioned or was implemented.

We think that this project has been difficult in the sense that there has not been any definite finish line work wise. There have been so many possibilities in this system that it sometimes was hard to focus on finishing one part before starting on the next. One of the things that we did not implement was the car pooling idea which we think would bring more value to the system.

8.3 Recommendations for further work

The idea for the system comes from the experiences of commuting in a larger city. Worth noting is that our observations of road-user behavior, commuting etc. is based on the traffic situation in Sweden, foremost the city of Gothenburg. Our implementation of the system was done for use in and around the city of Gothenburg. Trafikverket has different traffic management centers around Sweden and they are all responsible for their geographical area. This means that the data used in our implementation comes from the traffic management center in Gothenburg, which mainly covers the western parts of Sweden. The data from the other traffic management centers should follow the same pattern but there might be local modifications and this is something that needs to be investigated before feeding the system with that data.

As for the accelerometer the project team decided that the accelerometer data was too uncertain to be used in a car crash application. The project team only conducted limited testing of the mobile phone accelerometer; it would be interesting to test the mobile phone accelerometer in a car crash environment to see if it could be a useful tool for detecting car crashes.

Even if the system is already usable for traffic information it could always be improved with other services. Here is a list of some ideas that we think would be great possibilities for the future.

- One possibility would be to extend the messages that we send to the user to include a map of how he could avoid the traffic event so that he does not get stuck in traffic. Right now we give textual redirection instructions that we get from Trafikverket and they are not always available.
- A good feature would be to allow the data from the server to be imported from/exported to various standard formats for use in other applications. This would allow a user that has already set up his routes in another application to easily import and use them in our system. And by allowing the user to export his routes we allow him to use them in other programs.
- One could also extend the web interface to allow the user to see old and current events on a selected route right there in the browser. The user might also like to see what happens at an arbitrary location in the country and we could present that also, by allowing the user to click on a location on the map.
- Car pooling is another possible extension. Users who drive the same route might allow others to see this and contact them to suggest car pooling if possible. By having this in the web interface users who might not otherwise approach someone about car pooling could find it easier through this interface. There is however privacy concerns with this that needs to be taken into consideration.
- Users might not want to create routes for everything and might be satisfied with being notified of new events that are close to their current location. This could be achieved by having the Android application report the user's current location to the server which could then check if there are any traffic events within a set distance.
- One could extend the system to allow for users to not only create routes but also allow them to create areas (polygons) that they are interested in receiving traffic information about. This could be interesting for, for example, journalists or photographers that could use it to be on the scene

quickly. They could receive a SMS when a new traffic incident has happened in their defined area. This could perhaps be something that they are willing to pay a monthly/yearly fee for.

Another idea might not be further work for this system but instead more like a similar system. The idea is to let users choose which geographical areas that they want to receive information in. You define your desired area by drawing a polygon on a map. This could then be used by business owners to deliver targeted advertisement (special offers) to these users. Let's say a user registers and enters his name, age and gender. He will then define the area that he is interested in receiving advertisement from, and perhaps which categories of businesses that he is interested in. The advertisement could perhaps be sent out by SMS since most people probably would not want to have an application running only to receive advertisement.

A business owner could then register on the site as well and provide the location, and categories, of his business (perhaps pay a monthly/yearly fee). When an owner is interested in advertising something he could log on to the site and would directly be presented with how many people that would receive his advertisement. He could see which age groups and genders the target audience would have and perhaps filter out and only deliver the advertisement to those he thinks he would benefit the most from. For these advertisements there could also be a one-time fee per delivery recipient. If you can get the owners to have special offers that are only available to the users of this system, for example by having a unique code for the advertisement, you could perhaps generate a large interest in the service.

8.4 Final thoughts

We never intended to create a system that was ready for launch to the public. The whole idea with this master thesis was to see what was possible to do with traffic information and mobile phones. We wanted a system where users could create their own routes and then receive traffic information for those routes. We also wanted to create a system where the users could report their own observations and by doing so help in generating a better experience for all users.

Even though we did not set out to create a system ready for launch we did create a big part of what is needed for such a system. A user can register for an account, create/read/update and delete routes. He can select one of his routes in the mobile application and see the most recent traffic events along that route. The system will receive traffic events from Trafikverket and make sure that they are stored correctly in the database and also delivered to the routes that are connected to them. The mobile application is able to receive these routes and the user can also report his observations using the application. In the end this means that we have a functional, although quite rough, implementation of the most critical parts of the system.

9. List of references

Abdel-aty, M.A., Kitamura, R. & Jovanis, P.P., 1997. Using stated preferemce data for studying the effect of advanced traffic information on drivers' route choice. *Transportation research Part C: Emerging Technologies*, 5(1), pp.39-50.

Abran, A. et al., 2004. *SWEBOK*. Los Alamitos: IEEE Computer Society.

Arnott, R., de Palma, A. & Lindsey, R., 1991. Does providing information to drivers reduce traffic congestion. *Transportation Research Part A: General*, 25(5), pp.309-18.

Barisani, A. & Bianco, D., 2007. Unusual Car Navigation Tricks: Injecting RDS-TMC Traffic Information Signals. In *CANSEC West/core07*., 2007.

Bellavance, F., 2005. Linking data from different sources to estimate the risk of a collision when using a cell phone while driving. In *International Conference on Distracted Driving*. Toronto, 2005.

Ch.6, P., n.d. *Chapter 6. Performance Tips*. [Online] Available at: <http://postgis.refractions.net/docs/ch06.html#id2802268> [Accessed 28 May 2010].

Chouayakh, S., 2007. *The mobile network as a traffic information tool*. Göteborg: Chalmers University of Technology.

code.google.com, 2010. *Google Web Toolkit*. [Online] Available at : <http://code.google.com/webtoolkit/> [Accessed 28 May 2010].

Colquhoun, S., 2008. *Anatomy of a crash*. [Online] Available at: < http://www.drive.com.au/Editorial/ArticleDetail.aspx?ArticleID=56781> [Accessed 28 May 2010].

DATEX II.eu, n.d. *DATEX II Background | DATEX II*. [Online] Available at: <http://www.datex2.eu/content/datex-background > [Accessed 23 May 2010].

developer.android.com, 2010. *Developing In Other IDEs - Android Developers*. [Online] Available at: <http://developer.android.com/intl/de/guide/developing/other-ide.html> [Accessed 28 May 2010].

Google Maps, 2010. [Online] Available at http://maps.google.com [Accessed 28 May 2010]

hibernate.org, n.d. *Hibernate - JBoss Community*. [Online] Available at: <http://hibernate.org/> [Accessed 28 May 2010].

hibernatespatial.org, 2010. *Hibernate Spatial*. [Online] Available at: <http://www.hibernatespatial.org/> [Accessed 28 May 2010].

Holpers, J. & Lindh, M., 2009. *Informationsspridningen i samhället - Till vilken nytta för slutkunden*. Göteborg: Handelshögskolan vid Göteborgs Universitet.

HTC Corporation, 2010. *HTC - Products - HTC Magic - Overview*. [Online] Available at: <http://www.htc.com/www/product/magic/overview.html> [Accessed 23 May 2010].

ISO International Organization for Standardization. 2008. Guide for ISO/IEC 12207 (Software Life Cycle Process).

Ketabdar, H., 2009. Detecting Physical Shock by a Mobile Phone and its Applications in Security and Emergency. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*. Bonn, Germany, 2009.

Mahmassani, H.S. & Liu, Y.-H., 1999. Dynamics of commuting decision behaviour under advanced traveller information systems. *Transportation research Part C: Emerging Technologies*, 7(2-3), pp.91-107.

Maylor, H., 2005. *Project Management*. 3rd ed. Essex: Pearson Education Limited.

McCormack, A., 2001. Why Evolutionary Software Development Works. *MIT Sloan Management Review*, 42(2).

mqtt.org, 2010. *MQ Telemetry Transport*. [Online] Available at: <http://mqtt.org/> [Accessed 18 February 2010].

Nelson, R.A., 1999. *The Global Positioning System - A National Resource*. [Online] Available at: <http://www.aticourses.com/global_positioning_system.htm> [Accessed 28 May 2010].

Olsson, Lennart. 2010. Personal Communication.

opengeospatial.org, 2010. *Simple Feature Access - Part 1: Common Architecture*. [Online] Available at: <http://www.opengeospatial.org/standards/sfa/> [Accessed 28 May 2010].

OpenStreetMap.org, 2010. *OpenStreetMap Wiki*. [Online] Available at: <http://wiki.openstreetmap.org/wiki/Main_Page> [Accessed 28 May 2010].

PostGIS, n.d. *PostGIS*. [Online] Available at: <http://postgis.refractions.net> [Accessed 10 April 2010].

Redelmeier, D.A. & Tibshirani, R.J., 1997. Association between cellular-telephone calls and motor vehicle collisions. *The New England Journal of Medicine*, 336(7), pp.453-58.

Sihvola, N. et al., 2009. In-depth evaluation of the effects of an automatic emergency call system on road fatalities. *European Transport Research Review*, 1(3), pp.99-105.

Thompson, C. et al., n.d. *Using Smartphones to Detect Car Accidents and Provide Situational Awareness to Emergency Responders*. Nashville, TN USA: Vanderbilt University.

tmcforum.com, 2004. *tmcforum.com : What is Traffic Message Channel (TMC)?* [Online] Available at: <http://www.tmcforum.com/en/about_tmc/what_is_tmc/> [Accessed 28 May 2010].

To, H. & Choudhry, O., 2000. *Mayday Plus Operational Test*. St. Paul, Minnesota: Minnesota Department of Transportation.

Trafikverket, 2009. *TMC - Traffic Message Channel*. [Online] Available at: <http://www.vv.se/Startsida-foretag/Service--e-tjanster/Trafikinformation/TMC---Traffic-Message-Channel/> [Accessed 4 May 2010].

vcu.edu, n.d. *Kinetic Energy*. [Online] Available at: <http://www.vcu.edu/cppweb/tstc/crashinvestigation/kinetic.html> [Accessed 28 May 2010].

Wikipedia, 2010. *Mobile phones and driving safety*. [Online] Available at: <http://en.wikipedia.org/wiki/Cell_phone_bans> [Accessed 28 May 2010].

Yee, M., 2003. *Acceleration That Would Deploy Car Airbags*. [Online] Available at: <http://hypertextbook.com/facts/2003/MichelleYee.shtml> [Accessed 28 May 2010].

10. Appendices

Appendix A - Project plan
Appendix B - Time plan
Appendix C - System Description v.1
Appendix D - System Description v.2
Appendix E - Survey result
Appendix F - Web interface SRS
Appendix G - Mobile application SRS
Appendix H - Android application GUI presentation
Appendix I - Communication document

Appendix A
Project plan
"Building a traffic information system"

Table of contents

A1. Client

The clients for this thesis are Chalmers University of Technology and Crepido Systems. Crepido Systems is a software development company with a local office in Gothenburg.

A2. Overall project description

This project is a master thesis done at Chalmers University of Technology. The idea for the master thesis comes from Michael Walenius at Crepido Systems in Gothenburg. Michael has a concept idea about a computer software that handles traffic information. The concept is very broad and our objective is to research this concept to see what works and what does not and in the end come up with a system design. Apart from designing the traffic information system the objective with the master thesis is also to deliver mobile-phone software that detects car-crashes. The mobile-phone software will be based on the Android platform.

A2.1 Purpose

The purpose of this project is to deliver a system design for a traffic information system and to implement a proof-of-concept application for the Android mobile platform. The traffic information system will deliver detailed traffic information from reliable sources to users in a way that best suits their needs. This could either be through a SMS message, rss-feed or a standard website. This will allow users to have up-to-date traffic information available when they need to make choices about what routes to take.

The other part of this project will be to implement a proof-of-concept application for the Android mobile platform. This application will test if it is possible to create an application that automatically contacts the emergency services in case of a car crash. The application will detect a crash using the accelerometer of a mobile phone and then automatically send a message informing the emergency services of the accident (with information such as phone number, location, if available, and force measured) and if they call back the phone will automatically answer and put it on speaker phone.

A2.2 Deliverables

This project will have an overall time plan that can be seen in appendix B. The project will be planned in detail in five week stages. Every five weeks the project team will take one week to document the project, write on the thesis report etc. During this week the team will also sum up the project so far and plan the next five weeks in detail. This means that this list of deliverables lists the major deliverables and each deliverable might in the end deliver several documents.

The project deliverables are:

- Project plan
- System description
- Concept breakdown
- Research and limitations document
- System design documents
- Android application documentation
- Master thesis report

A2.3 Demarcations

The project demarcations will be finished when the "Concept scope analysis and breakdown" phase is completed and presented in the "Research and limitations document".

A3. Phases

The project will consist of six phases containing the following key activities. As earlier mentioned project details are only planned five weeks at the time. This means that the later phases do not have a detailed plan yet.

A3.1 The pre-project phase

- Getting the project approved by Chalmers
- Project Planning

A3.2 Concept scope analysis and breakdown

- Concept brainstorming
- System description
- Concept breakdown

A3.3 Research and limitations

- Research the different system aspects
- System limitations and demarcations

A3.4 System design

- Details about this phase will be decided at a later point when the research is completed.

A3.5 Android

- Examine possibilities and limitations in the Android technology.
- Software design
- Implementation and testing of software

A3.6 Documentation and finalizing report

- Finalizing the project report
- Project evaluation
- Project presentation

A4. Project Organization

The project only has two members, Magnus Jonsson and Mattias Svensson. There are no specific responsibilities such as project manager in this project. Delegation and division of work will be done by both project members as needed.

A5. Documentation plan

Document	Approved by	Purpose	Distributed to	Deadline
Project plan	Per Zaring and Michael Walenius	Defines the purpose of the project.	Per Zaring and Michael Walenius	2010-01-24
System breakdown	Per Zaring and Michael Walenius	Presents the system concept.	Per Zaring and Michael Walenius	2010-02-07
Limitations and demarcations	Per Zaring and Michael Walenius	Presents the limitations and demarcations of the system.	Per Zaring and Michael Walenius	2010-02-21
System design	Per Zaring and Michael Walenius	Presents the traffic information system design.	Per Zaring and Michael Walenius	2010-05-02
Android	Per Zaring and Michael Walenius	Presents the proof-of-concept Android application.	Per Zaring and Michael Walenius	2010-05-16
Final report	Per Zaring	This report will present the entire project and will also be the final report of the master thesis.	Per Zaring, Michael Walenius and Chalmers University of Technology	2010-06-13

A6. Report and meeting routines

A monthly status update will be sent to Per Zaring to keep him up-to-date with the project progress. The project team will have continuous contact with Michael at Crepido Systems and he will get status report of project progress every 14 days.

A7. Resource allocation plan

A7.1 People

Both members of the project group will be fully available during this project according to the time plan.

A7.2 Equipment

The project team will use their own laptops which gives the project great flexibility. Monitors and other equipment will be available for use at Crepido Systems office.

A7.3 Facilities

The work will be conducted both at Chalmers, Crepido Systems and in the homes of the project members.

A7.4 Finance

There is no financial budget available for this project.

A8 Milestones

#	Description	Date
1	Project plan completed	2010-01-24
2	Concept scope analysis and breakdown completed	2010-02-07
3	Research and limitations completed	2010-02-21
4	System design completed	2010-05-02
5	Android completed	2010-05-16
6	Report finished	2010-06-13
7	Thesis finished, Hello World!	2010-06-13

A9 Time plan

The time plan can be seen in appendix B.

10 Risk analysis

Since this is a project without a budget and with no critical deadlines the risks of the project are small. The largest risk in the project is critical data loss and to avoid data loss the following actions has been taken.

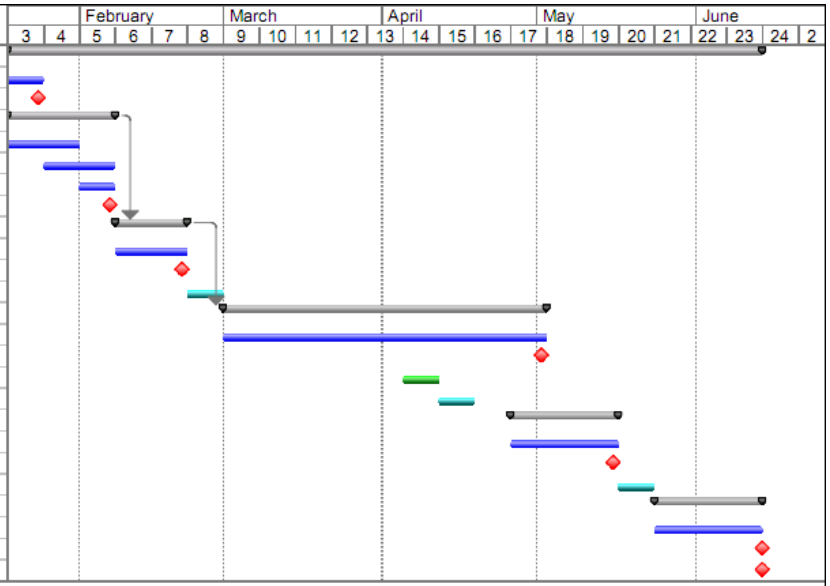
- A majority of the documentation is saved in Google Docs which has a low risk of failure.
- Regular backups of all project data will store the project data on five different storage media at three different locations.

This project only contains two members and the project team will work together on all documents. This means that both members of the team will be fully informed of the project at any given point which reduces the severity of project staff fall of or long term illness of project member.

Activity	**Severity**	**Likelihood**	**Total**
Critical data loss	8	4	32
Long term illness of project member	6	2	12
Project staff fall off	9	1	9
Crepido Systems or Michael Walenius leaves project	5	1	5

Appendix B
Time plan

Name	Duration	Start	Finish
Project Total	**108 days**	**Mon 10-01-18**	**Sun 10-06-13**
Create project plan	6 days	Mon 10-01-18	Sun 10-01-24
M1: Project plan completed	0 days	Sun 10-01-24	Sun 10-01-24
Concept scope analysis and breakdown	**17 days**	**Mon 10-01-18**	**Sun 10-02-07**
Concept brainstorming	11 days	Mon 10-01-18	Sun 10-01-31
System description	11 days	Mon 10-01-25	Sun 10-02-07
Concept breakdown	6 days	Mon 10-02-01	Sun 10-02-07
M2: Concept scope analysis and breakdown comp	0 days	Sun 10-02-07	Sun 10-02-07
Research and limitations	**11 days**	**Mon 10-02-08**	**Sun 10-02-21**
Research the different system aspects, limitations	11 days	Mon 10-02-08	Sun 10-02-21
M3: Research and limitations completed	0 days	Sun 10-02-21	Sun 10-02-21
Buffer for documentation, summary and replanning	5 days	Mon 10-02-22	Sun 10-02-28
System Design	**45 days**	**Mon 10-03-01**	**Sun 10-05-02**
System Design	45 days	Mon 10-03-01	Sun 10-05-02
M4: System Design completed	0 days	Sun 10-05-02	Sun 10-05-02
Easter break	5 days	Mon 10-04-05	Sun 10-04-11
Buffer for documentation, summary and replanning	5 days	Mon 10-04-12	Sun 10-04-18
Android	**15 days**	**Mon 10-04-26**	**Sun 10-05-16**
Android	15 days	Mon 10-04-26	Sun 10-05-16
M5: Android completed	0 days	Sun 10-05-16	Sun 10-05-16
Buffer for documentation, summary and replanning	5 days	Mon 10-05-17	Sun 10-05-23
Documentation and finalizing report	**15 days**	**Mon 10-05-24**	**Sun 10-06-13**
Documentation and finalizing report	15 days	Mon 10-05-24	Sun 10-06-13
M6: Report finished	0 days	Sun 10-06-13	Sun 10-06-13
M7: Thesis finished Hello World!	0 days	Sun 10-06-13	Sun 10-06-13



Appendix C
System description v.1
Project TIS

Table of Contents

C1. General system description

The traffic information system is a system that helps road-users by making sure that they always have up-to-date information about the current traffic condition along their route. The user gets information about the current situation from a central server that communicates with a mobile phone application on the user's mobile phone. Apart from the mobile application the system also consists of a web page where the users can change their settings, create routes etc. The web page also presents information about the current traffic situation.

For this system to be of any use it is critical that it has as much data sources as possible. It is also important that it is easy to provide the system with data to make it attractive for data providers.

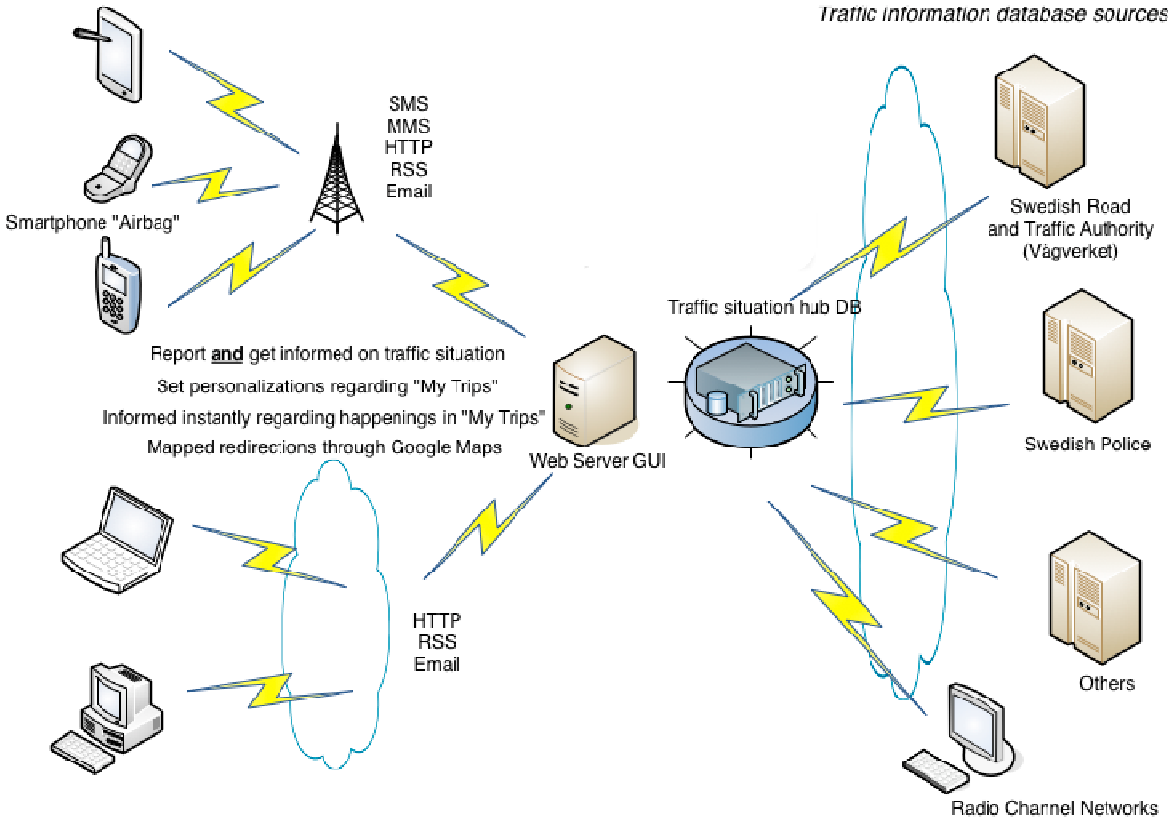


Figure 1. The system idea.

C2. Applications

The system has a number of different applications. The main application of the system is to provide road-users with current information about traffic conditions. This is described in part one and part two of the mobile application. The third mobile application is an application that will assist the user in case of a car crash. The system will also have a web interface which gives an overview of the current situation but will also be the place where the users can change their personal information and work with their routes. These are the main features of the system but apart from this the system could also be used as a hub for car

pooling. The users already have their different routes saved and if they also provide information about when they are planning to make these trips the system could be used to assist car pooling.

C2.1 Web

The web interface is intended to be the place where the user configures his personal settings and mobile application settings. It is here that the user enters his personal information and creates the routes that he wants traffic information about. From the web interface the user will also be able to import and export saved routes for other uses. When the user starts the mobile application it will synchronize personal information, route information etc. with the web interface.

Via the web interface the user should be able to change the following settings for the mobile application:

- How the information should be shown.
- For how long time the traffic messages should be shown.
- From which data sources messages should be presented.
- Traffic message update frequency.
- Via which technology to receive the traffic information.
- Change password.

Other settings include:

- Decide who can see the personal info.
- Change personal information.
- Turn the crash detection feature on/off.

C2.2 Mobile application

One part of the system is the mobile application which has three different parts.

C2.2.1 Part 1

The first part of the mobile application is intended to provide the user with traffic information along the route that he travels. When a user is about to begin his travel he can start the application and select a route that he has previously created in the web interface and quickly get a list of all the latest happenings along this route. If the user has not created the route he is about to drive beforehand, in the web interface, he should be able to create a new one on the fly.

As the user begins driving he can place the phone to aside and it will notify him of new events along his route as soon as they occur. When a new event occurs a sound will notify the user and a screen will display what has happened (this could also be read to the user by using text-to-speech to minimize the distraction since the user is now driving). If available the user will also get a description of what route to take to avoid getting stuck in traffic.

C2.2.2 Part 2

The second part of the mobile application can be used by the user when he is on the road to report accidents along the route, so that other drivers can become aware of the accident and perhaps avoid it. This should be able to be done in an easy way to minimize distraction. One way to do this could be by using the built in GPS (if available) for positioning and then have a predefined list of the most common

types of accidents to choose from, for example a car crash or a collision with an animal (treacherous road conditions could also be reported). By making this task easy more people could report accidents and thus giving the system a more complete look of how the roads are at any given point.

C2.2.3 Part 3

The third part of the mobile application is an autonomous system to help the user when an accident does occur. In case of a crash the phone is likely to be out of reach for the user or he might not be conscious and unable to call for help. The idea is that the phone will detect the car crash by using it's built in accelerometer and notify the emergency services with details of the crash, such as location, and the user's personal information. The emergency services can then call the user and the phone will automatically answer and put it on speaker phone so that the user can talk to them even if the phone is out of reach. When a crash is detected the phone can also display information about the user on the screen to make it easier for people arriving to the scene. This information could be the user's name, people to contact (to let them know that he has been in a crash) and necessary medical facts, such as allergies to medicine. This should help people coming to the scene to talk to the user and avoid giving him any medicine that might make him worse.

C2.3 Car pooling

Each user account contains one or several saved routes. By making these routes public and adding information about when the user will drive this route users can (if they want to) car pool with each other. By making routes public users can also see how other users travel from A to B and get useful tips.

C3. Data Sources

The system should be able to collect data from several different sources that have information about the current traffic conditions. The more data sources the system has the better information about current traffic conditions can be provided to the users. This section will suggest a number of different data sources.

C3.1 SOS

The Swedish SOS has all the information about the emergency services call outs. This includes call outs to the police, fire department and ambulance which all affect the current traffic situation in different ways. When an accident leads to traffic disturbances SOS could report this to the system and the system could notify the road-users. This would result in road-users having the ability to take alternative routes instead of getting stuck in traffic jams.

C3.1.1 The police

The police turns out on a great variety of events and some of these events could affect the current traffic situation. One such event could be traffic accidents. The police could benefit from having a way of notifying the road-users of disturbances in traffic and be able to divert traffic to avoid traffic jams. The Swedish police often announce on their web page on which roads they will set up speed controls, this is something that also could be included in the system.

C3.1.2 Fire department

The fire department turns out on traffic accidents and fires where they need to close parts of a road or sometimes entire roads or streets. By reporting these disturbances to the road-users the road-users can take action and choose alternative routes thus avoiding traffic jams.

C3.2 Users

The users of the system are very important since they are the ones on the road and will probably be the first ones to spot any disturbances in traffic. Apart from obvious disturbances such as accidents which the emergency services also reports the users can report about wild animals on the road, icy or snowy roads etc. the users have the advantage that they are the first ones on scene and they will therefore be one of the most important data sources in the system.

C3.3 The national road administration a.k.a "Trafikverket"

Trafikverket has a lot of information about the current traffic situation. They report to radio via their TMC system and they also have a web page (http://www.trafiken.nu/) where users can check the current traffic situation. This information would be ideal to have in the system. Another type of road disturbances is road work which Trafikverket also has information about.

C4. Data stored

C4.1 Personal information

The personal information stored is primarily for use in the car crash detection part of the mobile application. The following information should be stored.

- Name.
- Phone number.
- Personal ID number.
- Address.
- Next of kin contact information.
- Medical facts.

C4.2 Route information

The user should be able to create and save routes and each user should be able to have several different routes. The user should get an estimate on how long time each route will take. A feature for importing and exporting routes from/to popular formats would be desirable. If possible the user should be able to use his routes in a GPS. The user should be able to share their routes with other users and also be able to save other users routes. Each route should also be able to contain information about when it is taking place and if it is available for car pooling.

C4.3 Traffic events

Traffic events can be created either by some organization or by other users of the system. These events will contain information about who created the event, for example the police, and information like the location of the event (which can be supplied through the built in GPS of the phones when created by other users) and the time of the event. Besides that, the event will also contain a description of what has

happened (chosen from a predefined list of event types or entered manually) and, if it is known, how long it will take to clear up.

An event should have the following properties.

- Location (preferably GPS but some other way of reporting location is also needed).
- Time for event.
- Type of event.
- Estimated length (time until state is normal again).
- Severity (low, medium and high).
- Information about who this report came from (user, police, radio, fire etc).

C5. Technology

The user can choose how to be informed of new events, this could be either through a SMS message, an RSS-feed, e-mail or the through the mobile application connected with the system. A possible improvement could be to be able to update the route in a GPS to avoid getting stuck in traffic. This means that the system needs different ways of getting the information out to the users. It needs to be able to both send SMS messages to all operators and push information to the mobile application.

The system needs a way for users to be able to create and edit their routes. A possible way to achieve this can be to use the Google Maps API and connect that to the system. This could possibly allow users to use the standard way of getting directions for driving, and then alter them if they need to.

When it comes to the encryption part the system will rely on well established libraries for handling encryption of data and communication.

It is also important that the system is modular to make any future alternations easy. The system should also scale well, meaning that it should be able to handle 10 users as well as 10000 users.

C6. Security

As the intention is that this system should handle some personal information security becomes a big issue. The information needs to be encrypted both on the server and in the phone to avoid a third party getting hold of it. The idea is that the information should be edited in the web interface and then synchronized to the phone. This synchronization process also needs to be encrypted to avoid eavesdropping. By not allowing the personal information to be edited on the phone the security could be made tighter and the information should only be shown on the phone when synchronization is performed to check that all went right, and this should require some form of credential, typically a user name and password.

Appendix D
System description v.2
Project TIS

Table of Contents

D1. General system description

The traffic information system is a system that helps road-users by making sure that they always have up-to-date information about the current traffic conditions along their route. The user gets information about the current situation from a central server to an application on his mobile phone. Apart from the mobile application the system also consists of a web page where the users can change their settings, create routes etc. This webpage will, in our version, be a simple demonstration of what could be done for a complete system.

D2. Limitations and demarcations

After our research we have found that most data is actually available from a single source and we do not need to combine different data sources. Instead we will need to focus our attention on retrieving the data from one source, Trafikverket. Trafikverket has a rather complicated protocol for retrieving their data, called DATEX II, which will take some time to set up. We have also decided not to focus that much on the design of the interfaces since that requires performing tests with a selected group of people representing the general population; something which we do not have time for, or a great deal of training in.

D3. Applications

The system has a number of different applications. The main application of the system is to provide road-users with current information about traffic conditions. This is described in part one and part two of the mobile application. The third mobile application is an application that will assist the user in case of a car crash.

The system will feature a simple web interface for selecting routes that the user can store. These routes are then selectable from the mobile application when the user is about to begin his journey. The user will then get updates on traffic information along the chosen route as he drives.

The system will also need to store personal information for syncing to the mobile application in case the user is in an accident. This information will be optional and encrypted to keep it from prying eyes and it will only be shown on the mobile phone if a car crash has been detected.

D3.1 Web

The web interface will not be the main focus when we design our test system. It will be a simple interface to help to test and demonstrate features of the system. The features that do need to be implemented are the ones to create and save routes so that the mobile application has something to select from when it starts and the system has something to check traffic information against to see what needs to be pushed to the user’s phone.

D3.2 Mobile application

One component of the system is the mobile application which consists of two different parts.

D3.2.1 Part 1

The first part of the mobile application is intended to provide the user with traffic information along the route that he travels. When a user is about to begin his travel he can start the application and select a route that he has previously created in the web interface and quickly get a list of all the latest happenings along this route. If the user has not created the route he is about to drive beforehand, in the web interface, he should be able to create a new one on the fly.
As the user begins driving he can place the phone to the side and it will notify him of new events along his route as soon as they occur. When a new event occurs a sound will notify the user and a screen will display what has happened (this could also be read to the user by using text-to-speech to minimize the distraction since the user is now driving). If available the user will also get a description of what route to take to avoid getting stuck in traffic.

D3.2.2 Part 2

The second part of the mobile application is an autonomous system to help the user when an accident does occur. In case of a crash the phone is likely to be out of reach for the user or he might not be conscious and therefore unable to call for help. The idea is then that the phone will detect the car crash by using it's built in accelerometer and send notify the emergency services with details of the crash, such as location, and the user's personal information. The emergency services can then call the user and the phone will automatically answer and put it on speaker phone so that the user can talk to them even if the phone is out of reach. When a crash is detected the phone can also display information about the user on the screen to make it easier for people arriving to the scene. This information could be the user's name, people to contact to let them know that he has been in a crash and necessary medical facts, such as allergies to medicine. This should help people arriving on the scene to talk to the user and avoid giving him any medicine that might make him worse.

D4. Data Sources

The system will retrieve traffic information from the national road administration a.k.a Trafikverket. They have a lot of different data that covers everything from the weather to serious accidents. The protocol for retrieving this is called DATEX II and uses XML to format a request and get the result. This data is what will be presented to the user, but in a more user friendly way.

We originally had intended to approach SOS and ask if they would be willing to share their data, but it seems that they do share most of it with Trafikverket already (Olsson, 2010). This means we have most of the data that we wanted to have at the start of the thesis.

D5. Data stored

D5.1 Personal information

The personal information stored is primarily for use in the car crash detection part of the mobile application. The following information should be stored:

- Name.
- Phone number.
- Personal ID number.

- Address.
- Next of kin contact information.
- Medical facts.

D5.2 Route information

The user should be able to create and save routes and each user should be able to have several different routes. The user should get an estimate on how long time each route will take. The routes will be used to check if a traffic event lies on the route a user has selected to drive.

D5.3 Traffic events

Traffic events will be retrieved from Trafikverket. These events will contain information about who created the event, for example the police, and information like the location of the event and the time of the event. Besides that the event will also contain a description of what has happened and, if it is known, how long it will take to clear up.

An event should have the following properties.

- Location (preferably GPS but some other way of reporting location is also needed).
- Time for event.
- Type of event.
- Estimated length (time until state is normal again).
- Severity (low, medium and high).
- Information about who this report came from (user, police, radio, fire etc).

D6. Technology

The system needs a way for the users to be able to create and edit their routes. A possible way to achieve this can be to use the Google Maps API and connect that to the system. This could possibly allow users to use the standard way of getting directions for driving, and then alter them if they need to, to create their routes.

When it comes to the encryption part, the system will rely on well established libraries for handling encryption of data and communication.

It is also important that the system is modular to make any future alternations easy. The system should also scale well, meaning that it should be able to handle 10 users as well as 10000 users.

D7. Security

As the intention is that this system should handle some personal information security becomes a big issue. The information needs to be encrypted both on the server and in the phone to avoid a third party getting hold of it. The idea is that the information should be edited in the web interface and then synchronized to the phone. This synchronization process also needs to be encrypted to avoid eavesdropping. By not allowing the personal information to be edited on the phone the security could be made tighter and the information should only be shown on the phone when synchronization is performed to check that all went right, and this should require some form of credential, typically a user name and password.

Appendix E
Questionnaire report
Project TIS

Table of Contents

E1. Summary

This questionnaire was created to give input to the preliminary system description. The questionnaire consisted of 12 questions. Of the participants 90 % were between 21 and 35 years old and 78% of them were male. This distribution is not ideal and should be taken into consideration when looking at the survey question results. The results show that it is very important that the system is fast and safe for the road-user to use. The traffic messages should preferably be presented in text and it is also important that the system works in larger parts of Sweden.

E2. Background

This questionnaire was done to give input on the development of a traffic information system. The traffic information system is a system that helps road-users by making sure that they always have up-to-date information about the current traffic conditions along their route. At the point when this questionnaire was made a general system description had been done and the purpose of the questionnaire was to get user input about the general idea behind the system.

E3. Questionnaire procedure

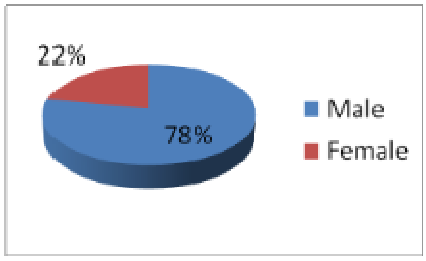
The questionnaire had 12 questions and was designed to give input about the general system idea. The first question was about the participants' age and gender to be able to see if the questionnaire had a good participant variance. After these initial questions the questionnaire was split into two main parts, one about the main system and the other about the mobile application connected with the system.

E3.1 Results

The respondents of the questionnaire have been friends and friends of friends which mean that the answers might be somewhat partial. The questionnaire was sent to 55 people and got 41 responses, which is a return rate of 75%; a questionnaire normally receives a return rate of 25-30%[1]. The unusual high answer rate might be because of the fact that the questionnaire was internet-based and had a low number of questions. However, the most likely reason is that the questionnaire was mostly sent to friends of the project-team and that the participants therefore felt more obliged to respond.

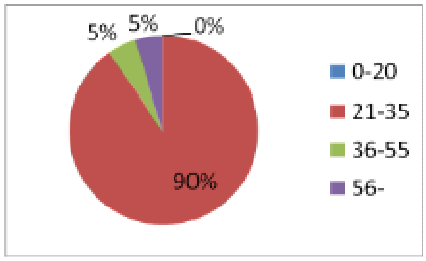
E3.2 Respondents

The first two questions were general questions about the participants' gender and age.



As seen in this first chart most of the people who answered the survey where male.

[1] Dix, A et.al. 2004. *Human-Computer Interaction*. 3rd ed. Essex. Pearson Education Limited. pp 351

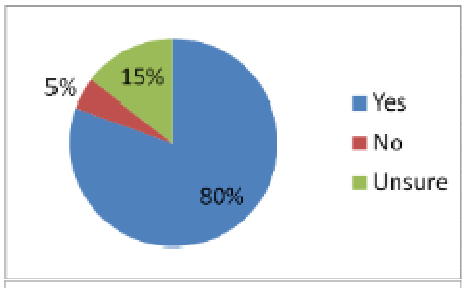


90 % of the participants are in the age group 21-35. The result from the first two questions show that the respondents cannot be seen as a representation of what the general population would think of this system. The questionnaire lacks answers from older people, who often do not have the same usage patter as young people who have grown up with mobile phones and internet. The ideal situation would have been a more even distribution both age and gender wise.

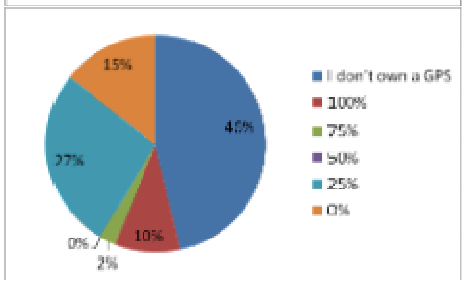
E3.3 System

A general description of the system was presented and the participants were asked questions about their thoughts of the system, such as how they ranked the importance of different features. Our description of the system was as follows:

"The traffic information system will help road-users by supplying them with current traffic information. To make sure the information is as current as possible the system will use a number of different data sources, like the users themselves reporting traffic events. Every road-user can get information on the general traffic situation or about the situation on the route they are about to travel. By giving road-users continuous information on the traffic situation we hope that the system can give road-users a possibility of taking alternative routes before they get stuck in traffic because of an accident, slippery roads etc."

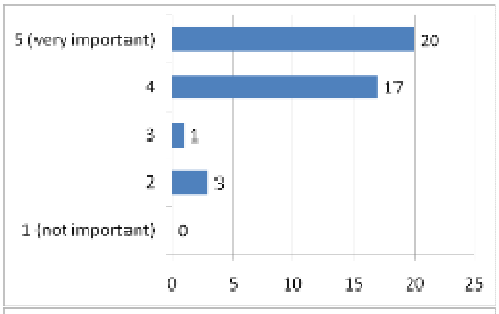


After the participants were given the general description of the system they were asked if they would use such a system. The result was that most participants could consider using the system.

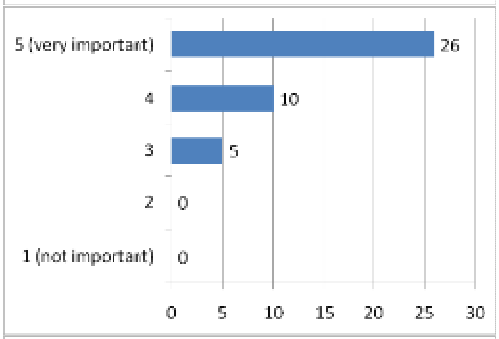


To see if the participants already used some sort of aid when driving the participants were asked how often they used a GPS while driving, if at all. The result was that nearly half of the participants do not actually own a GPS and the ones that do, are not using the GPS very often.

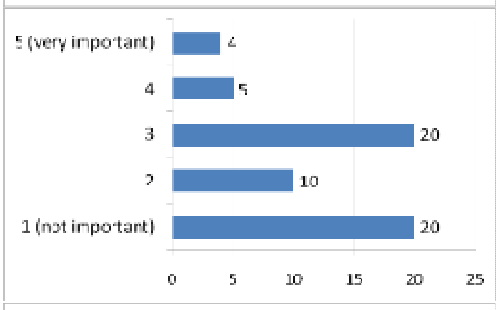
When the more general questions had been asked the focus shifted to ask the participants about how important some specific features of the system were. The main question was "What would be important for you in such a system?" and the participants were then asked to rank six statements from one (not important) to five (very important).



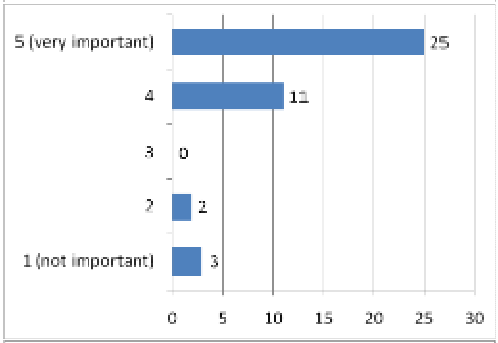
The first statement was “That you as a road-user gets an alternative route in case of a traffic incident.” The result shows that most people rank this as a very important feature.



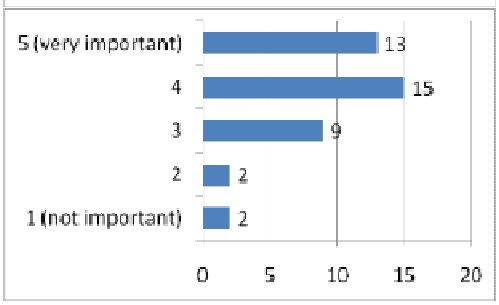
The second statement was “That information about traffic incidents is available fast.” The project-team wanted to know how important the participants felt that it was for the information to get to them quickly. The results show that most people found this important.



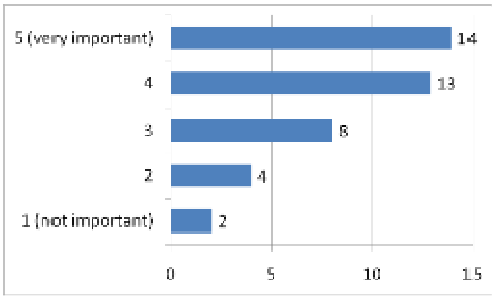
The third statement was “That you get information from a number of different sources.” This was a feature that most participants did not find important.



The fourth statement was “That the system works in large parts of Sweden.” Most participants found it important that the system covers large parts of Sweden.



The fifth statement was “That the system works in your city.” This statement is similar to the previous one; however, together with the previous statement it gives a better idea of where the participant would like to use the proposed system. This statement was important for the participants but not as important as the statement that the system works in most parts of Sweden.

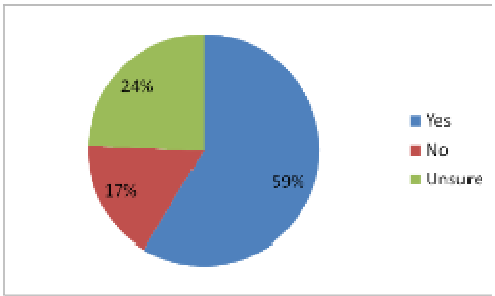


The sixth and last statement was “That it is easy for users to send in their own observations.” Since the preliminary system idea will depend on user traffic-reports it would be interesting to see if the participants found this important, which it was.

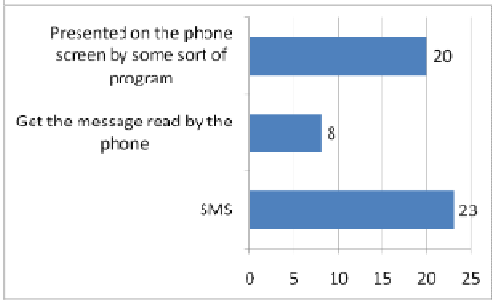
E3.4 Mobile phone

The idea for the mobile phone part of the system was presented with the following text:

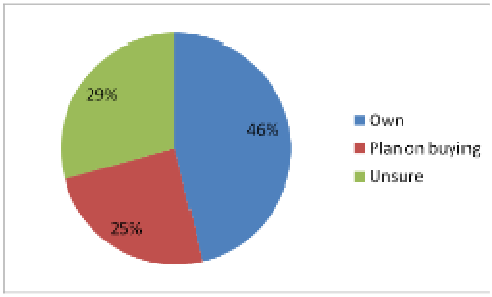
“To ensure that every road-user gets continuous updates while on the road the road-user should be able to get the updates sent to the mobile phone.”



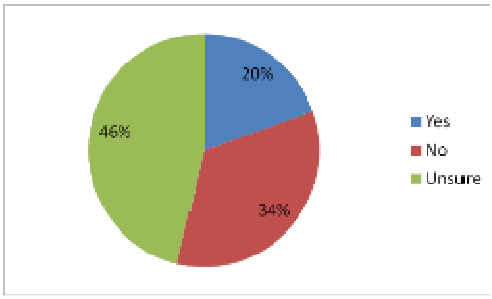
The first question was ”Would you use a solution like the one described above?” Most participants showed interest for a mobile phone solution.



The second question was regarding how the participants would like the information to be presented on the mobile phone. Most participants would like to get the information in text, either by SMS or by a specifically designed application The answer was a bit unexpected to the project-team since both of these solutions require the system user to take his eyes off the road.



The third question was “Do you own or plan to buy a smart phone.” A small majority did not own a smart phone.



The final survey question was if the participants would be prepared to pay for the system described in the questionnaire. A question that may not be so important from a system design view but very important from a business viewpoint. The result shows that a large part of the participants are not prepared to pay for the system or are unsure.

E3.5 Participant comments

The participants were given the opportunity to give general comments about the system after each part of the questionnaire. The first comments were to the open question “If you have suggestions on important properties/functions of the system you can list them below”. The comments to this statement can be seen below.

A big issue for the participants was that the system needs to be easy to use and not too technical. If it is difficult to enter information into the system users might skip it altogether. Automation came up as way to handle this. One other big concern that came up in a couple of responses was that of verification of user input. The participants were worried about for instance a user that enters false data to direct traffic away from his own route or in some other way sabotage for other users.

In case of a redirection from the system some participants were concerned that they might get redirected on some smaller roads and wanted a possibility change settings for this. Another idea was the possibility to only get notified of the location of the accident and that the user could decide for themselves if they wanted to change the route. One participant wanted to be able to change the frequency of the information updates, which is almost the opposite from another participant who wanted the information as close to real time as possible. Lastly one participant wants the system to use Google Maps and provide a public API for developing own solutions based on the data.

The next open question was “If you would like to get notified in a different way or have some other suggestions for the mobile phone application you can write them below.” and the answers are summarized below. The biggest concern when it comes to the mobile application was that of traffic safety, the application should not distract the user from their driving. The user should not have to do anything to get the information. Some participants said that they probably did not want a full screen solution but instead some sort of smaller notification. Some sort of sound notification to avoid having to look at the screen at frequent intervals was also a desired feature. Many participants wanted it integrated with the GPS instead of the mobile phone. One user also wanted an option to get the information as a RSS-feed to be able to use it on the computer also.

Finally the summary of responses to the open question “If you have any other comments you can write them below.” Some answers show that participants want the information available in other ways besides the mobile phone. Participants feel that there should be a free alterative and if they were to pay for a system like this it should be cheap and give benefits over the free versions such as more reliable sources and faster notification. One participant felt like this should be a public service driven by the government.

E4. Conclusion

90 % of the participants were between 21 and 35 years old and 78% of them were male. This distribution is not ideal and the answers are reflected by the lack of diversity in the participant group. Most of the participants did not have a GPS and those who had a GPS used it less than 25%. This conclusion can however also be the result of the fact that most of the participants are young males living in Sweden's second largest city i.e. they do not have a car. As for the system the questionnaire shows that it is important that the system is fast, can present alternative routes and that it works in large parts of Sweden (not just in the city). The presentation of the data on the mobile phone was preferred in text, especially SMS. The user comments showed concern about the traffic safety of this system and stressed the fact that it should be easy to use.

Appendix F
Software Requirement Specification
Web interface

Table of Contents

F1 Introduction

F1.1 Product Overview

The web interface will let the user configure his personal settings, routes, car pool information and some mobile application settings. Via the web interface the user should be able to control, create and edit all different kinds of data and get a good overview of the system and his account. As seen in figure 1 the web interface is one part of a larger system. The web interface role is primarily to let the user configure his routes.

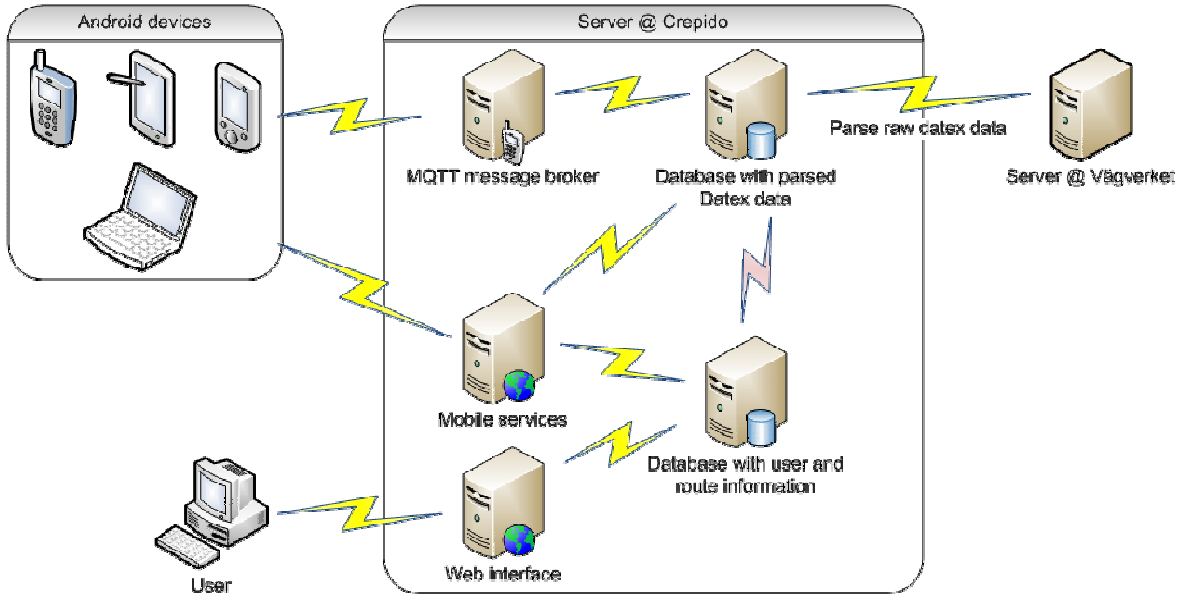


Figure 1. Detailed system overview.

F1.2 Aim and goal

The web interface is as mentioned one part of a larger system (see figure 1). The web interface is intended to be the place where the user makes all kinds of different user settings, it is therefore important that the web interface is easy to use. To have almost all settings in the web interface is important to eliminate confusion about where settings are made and to minimize data transfers from the mobile application. It is also much easier to edit settings on a computer than on a mobile phone which is limited in screen size and user input.

F2 Overview

F2.1 Product Description

It is important that the user can get a good and clear view of all different components and settings available. It is in the web interface that the user enters and edits his personal information and creates the routes that he travels frequently.

Route information
The user should be able to create and save routes, each user should be able to have several different routes. The saved routes will be the main component that decides which traffic information the user will get. The routes should be importable and exportable to various popular formats to extend the systems usability. Each user should be able to share his routes with other users and also be able to use other users' routes. Each route should also contain information about when it is taking place (for instance start time) and if it is available for car pooling.

Personal information
The user should be able to change his personal information and also manage routes and car pooling information. No user should be forced to share information and it should be possible for the user to set the privacy level.

Car pooling
Each user account will contain one or several routes. By making these routes public and adding information about when the user will drive the specific route the system will encourage users to car pool with each other. Another advantage of making the routes public is that users can see how other users travel from A to B and get useful tips.

F2.2 Dependencies with other systems

The web interface will communicate directly with a server both for retrieving and storing information such as settings and the personal information of the user. As the number of users grows a load balanced solution might be needed.

F2.3 Design philosophy

The system should be designed to allow for additional features to be added as required. It would also be a good decision to design the system to allow it to scale and be distributed among a number of servers.

F3 General system requirements

Functional

Number	Requirement	Priority
1.1	When creating an account the user should receive a message to his mobile phone to verify that it really is the user's phone.	High
1.2	The routes should be importable and exportable to various standardized formats.	Low
1.3	The user should get an estimate on how long time each route will take.	Low
1.4	A user should be able to recover a lost password.	High
1.5	The user should be able to log in via the web interface.	High

Non functional

Number	Requirement	Priority

1.6	The system should support a number of different languages.	Medium
1.7	The system should store each user's personal information.	High
1.8	The system should have administrator account(s).	High
1.9	The system should meet standards and laws for storing personal information.	High
1.10	The system should be expandable to handle a high number of users.	High
1.11	The system should have high availability.	High
1.12	The system should work in all major browsers.	High
1.13	Each user account should be associated with the user's phone.	High
1.14	Each route should contain a departure time stamp for car pooling.	Low
1.15	Each route should contain information regarding if it is available for car pooling or not.	Low
1.16	Each route should be made public and private (default private).	Low
1.17	The user's route should contain information about if the user wants to car pool that route or not.	Low
1.18	Deployment of the application should be easy and automatic.	High
1.19	The application should be extendable.	High
1.20	The application should be scalable.	High
1.21	The application should be testable.	High

F4 Specific Requirements

F4.1 User Interface

Functional

Number	Requirement	Priority
2.1	The web application should have a sign up function.	High
2.2	The user should be able to decide how the traffic information should be presented in the mobile application.	Medium
2.3	The user should be able to decide how long traffic messages should be shown on the mobile application.	Medium
2.4	The user should be able to decide how long traffic messages should be shown in the web application.	Medium
2.5	The user should be able to decide the traffic message update frequency on the mobile application.	Medium
2.6	The user should be able to decide the traffic message update frequency on the web application.	Medium

2.7	The user should be able to decide via which technology to receive the traffic information on the mobile application.	Medium
2.8	The user should be able to change password.	High
2.9	The user should be able to decide who can see his personal info.	High
2.10	The user should be able to update his personal information.	High
2.11	The user should be able to share his routes.	Low
2.12	The user should be able to see his routes on a map.	Medium
2.13	The user should be able to create routes.	High
2.14	The user should be able to edit routes.	High
2.15	The user should be able to save routes.	High
2.16	The user should be able to delete routes.	High
2.17	The user should be able to import and export his routes.	Low
2.18	The user should be able to search for other users for car pooling.	Low
2.19	The user should be able to browse other people's public routes.	Low
2.20	The user should be able to copy and use other people's public routes.	Low
2.21	The administrator should be able to delete users.	High
2.22	The administrator should be able to create users.	High
2.23	The administrator should be able to edit user information.	High
2.24	The administrator should be able to temporarily ban a user violating the system.	High
2.25	The administrator should be able to create new administrators.	High
2.26	The administrator should be able to upgrade existing users to administrator.	High
2.27	The administrator should be able to revoke administrator rights.	High

Non functional

Number	Requirement	Priority
2.28	The user interface should be easy to understand.	High

F4.2 Reliability

Non functional

Number	Requirement	Priority
3.1	The web interface should be available 24/7.	High

F4.3 Security

Functional

Number	Requirement	Priority
4.1	Information should be encrypted client side before it is sent to the server.	High

Non functional

Number	Requirement	Priority
4.2	The system should rely on tried libraries for security, no home cooked solutions.	High
4.3	Sensitive data should be encrypted.	High
4.4	The encryption should meet the requirements of rules and laws for storing personal information.	High
4.5	The data should be backed up at all times.	High

F5. Use Cases

User log in
Name: User log in
ID: UC-1
Actor: User
Goal: The user is logged in to the web interface.
Description: The user navigates to the web interface and inputs his username and password. The username and password is controlled and the web interface acknowledges that the login was successful.
Requirements covered: 1.5

Edit personal information
Name: Edit personal information
ID: UC-2
Actor: User
Goal: The user edits his personal information.
Description: The user is presented with a well structured list of his personal information and can edit this information.
Requirements covered: 2.8-2.10

Manage Routes
Name: Manage routes
ID: UC-3
Actor: User
Goal: The user deletes and creates routes.

Description: The user is presented with a view of his current routes. The user can then choose to edit, delete or create routes. Each route displays how long the route will take. The user should also be able to decide if he wants to share each route or not, and if it is available for car pooling.
Requirements covered: 2.13, 2.15, 2.16, 1.3, and 2.11

Edit Route
Name: Edit route
ID: UC-4
Actor: User
Goal: The user edits a selected route.
Description: The user is presented with a map where the selected route is displayed. The user can then change departure, destination and other variables of the route. Information about the travel time for each route is also displayed.
Requirements covered: 1.3, 2.12, and 2.14

Edit personal settings
Name: Edit personal settings
ID: UC-5
Actor: User
Goal: The user edits his personal settings.
Description: The user should be presented with a clear overview of all his different settings such as route settings, mobile application settings etc. that he can edit.
Requirements covered: 2.2-2.7

Create Account
Name: Create account
ID: UC-6
Actor: User
Goal: The user creates a new account.
Description: The user chooses to sign up and fills in all the necessary personal information and the phone number that should be associated with the account. The user then gets a confirmation message to the phone which the user needs to act upon to activate the account.
Requirements covered: 1.1 and 2.1

Administrate user accounts
Name: Administrate user accounts
ID: UC-7
Actor: Administrator
Goal: The administrator administrates accounts
Description: An administrator account should be available where the administrator can manage the user accounts. The administrator should be able to add-, edit-, delete user accounts, temporarily ban users, create new administrators, upgrade users to administrators and revoke other administrators rights.
Requirements covered: 2.21 - 2.27

Import and export route

Name: Import and export route
ID: UC-8
Actor: User
Goal: Import or export route.
Description: The user can import or export a route from/to popular formats.
Requirements covered: 1.2 and 2.17

Recover password
Name: Recover password
ID: UC-9
Actor: User
Goal: Recover a lost password
Description: If the user has lost his password he can fill out a recover password form where he enters his e-mail and a new generated password is sent to that e-mail address.
Requirements covered: 1.4

Browse public routes
Name: Browse public routes
ID: UC-10
Actor: User
Goal: Browse the public routes and if wanted, add routes to user's routes.
Description: The user can browse other users' routes and add these routes to his own routes. He can also search for routes that are available for car pooling.
Requirements covered: 2.19, 2.20, and 2.18

Appendix G
Software Requirement Specification
Mobile application

Table of Contents

G1 Introduction

G1.1 Product Overview

The mobile application will display traffic information and other types of information from other parts of the system to the user. As seen in figure 1 the mobile application receives its information from a mobile services server and the MQTT message broker.

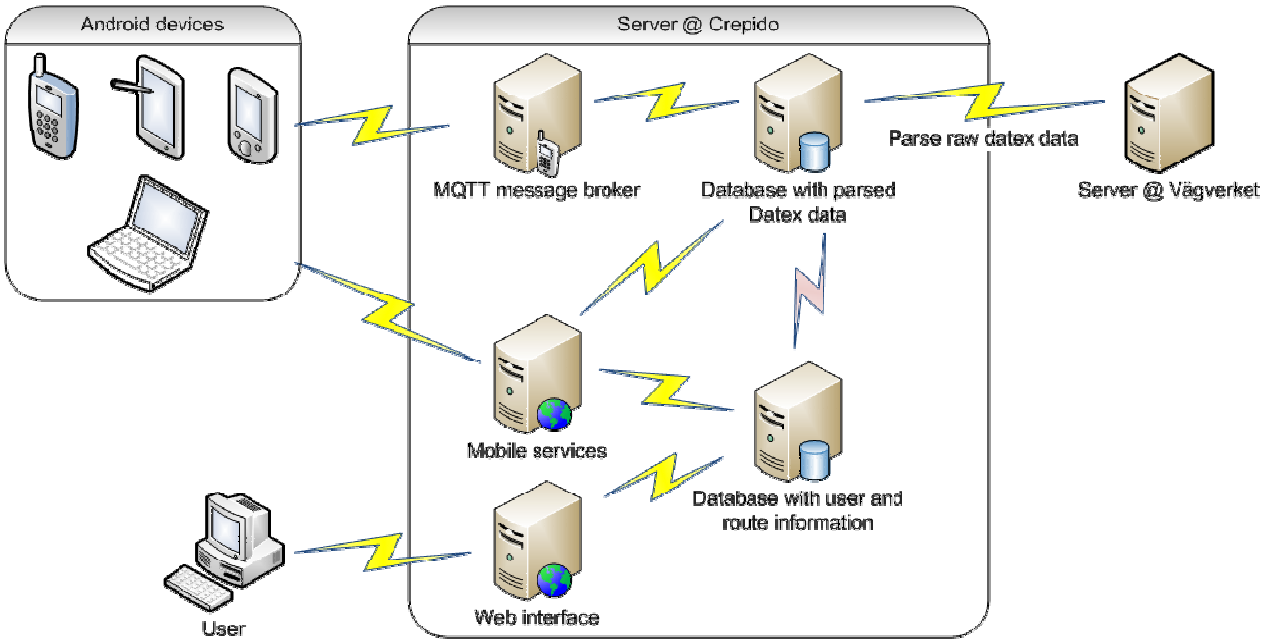


Figure 1. Detailed system overview.

G1.2 Aim and goal

The mobile application is intended to be the main communication link between the main server and the user. It is very important that the GUI of the mobile application is easy to use since it sometimes will be used while the user is driving. The user interface design is therefore highly important for this application. The design will be further discussed in G2.3.

G2 Overview

G2.1 Product Description

The mobile application will consist of three main parts, the start screen, the pre-route display and the on-route display, as seen in figure 2. The pre-route part of the application will give the user information about the most recent traffic messages on the selected route, current weather reports and a current road condition overview. The intention is that the user can take part of this information to aid the trip to his destination. When the user is driving the mobile application will be in on-route mode. This means that traffic messages affecting the selected route will be displayed and there will also be possible for the user to report traffic

messages back to the traffic information server. Other features like locating the nearest rest area will also be available to help planning breaks along the way.

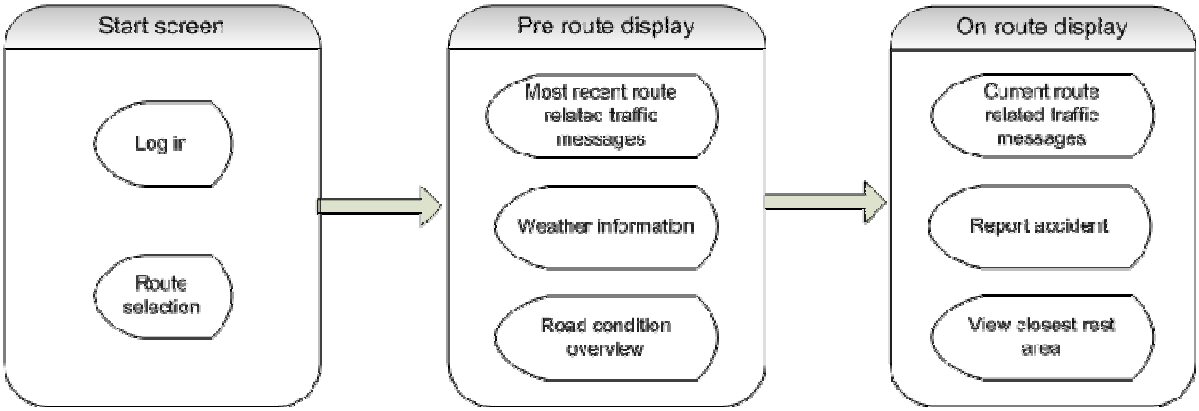


Figure 2. The mobile application flow diagram.

G2.2 Dependencies with other systems

The mobile application will establish two-way communication with the main server. Without the main server the mobile application will not be able to show any traffic information or user related data. There is also a connection to the webpage yr.no where the latest weather reports are downloaded from.

G2.3 Design philosophy

As earlier mentioned the mobile applications intention is to be used while on the road. It is therefore very important that the application does not require much user interaction and that it is easy for the user to understand the traffic information. The user interface design's key elements will be large text and an easy to understand layout.

G3 General system requirements

Functional

Number	Requirement	Priority
1.1	The application should automatically try to detect in which county the user is in.	Medium
1.2	If the application cannot detect which county the user is in the user should be able to choose this.	High
1.3	Information about rest areas and their position should be viewable on a map.	Medium
1.4	Traffic messages should be viewable on a map.	High

Non functional

Number	Requirement	Priority
1.5	The application should be testable.	High

1.6	Deployment of the application should be easy and automatic.	High
1.7	The application should keep a number of the latest traffic messages available for the user.	Medium

G4 Specific Requirements

G4.1 User Interface

Functional

Number	Requirement	Priority
2.1	The user should be able to set credentials for login.	High
2.2	The user should be able to choose among the different routes he has created.	High
2.3	The user should be able to view traffic messages related to the chosen route.	High
2.4	The user should be able to view a weather forecast for the user's current location.	Low
2.5	The user should be able to get a report of the current road conditions.	Medium
2.6	The user should be able to view information about rest areas close to the user's current location.	Medium
2.7	The user should be able to report different kind of accident types.	High
2.8	The user should be able to start the traffic message monitoring.	High

Non functional

Number	Requirement	Priority
2.9	The user interface should be easy to understand.	High
2.10	The user interface should be easy to navigate.	High
2.11	The user interface should require as little user interaction as possible.	High

G4.2 Reliability

Non functional

Number	Requirement	Priority
3.1	The application should be able to run in the background and alert the user when new messages arrive.	High

G4.3 Security

Functional

Number	Requirement	Priority
4.1	The login credentials should be encrypted client side before being sent to the server.	Medium
4.2	The user must log in to use the application.	Medium

Non functional

Number	Requirement	Priority
4.3	Sensitive data should be encrypted.	High
4.4	The encryption should meet the requirements of rules and laws for storing personal information.	High

G5. Use Cases

Log in
Name: Log in
ID: UC-1
Actor: User
Goal: The user should be authenticated and logged in when accessing the system.
Description: The user authenticates himself to the system using his credentials and is logged in to the system.
Requirements covered: 2.1, 4.1 and 4.2

Choose route
Name: The user selects one of his routes
ID: UC-2
Actor: User
Goal: A route is selected and this route will be used by the rest of the system.
Description: The user is presented with the routes he has created and selects one to use for this run of the application.
Requirements covered: 2.2

View traffic messages
Name: View traffic messages
ID: UC-3
Actor: User
Goal: The user is presented with the most recent traffic messages along the selected route.
Description: The user is presented with traffic messages, and their details, and can choose to display a traffic message on a map.
Requirements covered: 1.4 and 2.3

View weather forecast
Name: View weather forecast
ID: UC-4

Actor: User
Goal: The user is presented with a weather forecast for the selected route.
Description: The user is presented with an overview weather forecast and can choose to get more details.
Requirements covered: 2.4

View road condition overview
Name: View road condition overview
ID: UC-5
Actor: User
Goal: The user is presented with a road condition overview for the selected route.
Description: The application determines which county the user is currently in and displays a road condition overview for that county. If the application cannot determine the user's current county the user should be allowed to choose among the available counties.
Requirements covered: 1.1, 1.2 and 2.5

Start traffic message monitoring
Name: Start traffic message monitoring
ID: UC-6
Actor: User
Goal: The application should be connected to the system and ready to display new traffic messages to the user as they arrive.
Description: When the user selects to start the traffic message monitoring the application should connect to the system and keep that connection alive until the users decides to end the monitoring. When new traffic messages arrive they should be displayed to the user and the last number of messages should be available for the users to see again.
Requirements covered: 2.8

Report accident
Name: Report accident
ID: UC-7
Actor: User
Goal: An accident should be reported to the system by the user.
Description: A user is allowed to choose from a number of predefined types of accidents and how severe the accident is. This information is then reported to the system.
Requirements covered: 2.7

View closest rest area
Name: View closest rest area
ID: UC-8
Actor: User
Goal: The user is presented with a map of his location and the closest rest areas.
Description: The user is presented with a map of the closest rest areas and can choose to see more details for each one of them.
Requirements covered: 1.3 and 2.6

Appendix H
Android application GUI presentation

Table of Contents

H1. Android GUI overview

The Android applications graphical user interface is divided into three parts, log-in, pre-route and on-route. The login display lets the user choose route and set settings such as username and password. When the log-in credentials has been checked by the server the user is taken to the second part of the application, the pre-route display. The pre-route display is designed to give the user information about current traffic messages on the selected route, weather information and a road condition overview. With this information the user will be well prepared when he starts his travel. When the user has started to travel he switches to the on-route part of the application. The on-route display receives and displays traffic messages related to the user route. From here the user can also report accidents and view information about the closest rest areas. The application flow can be seen in figure 1.

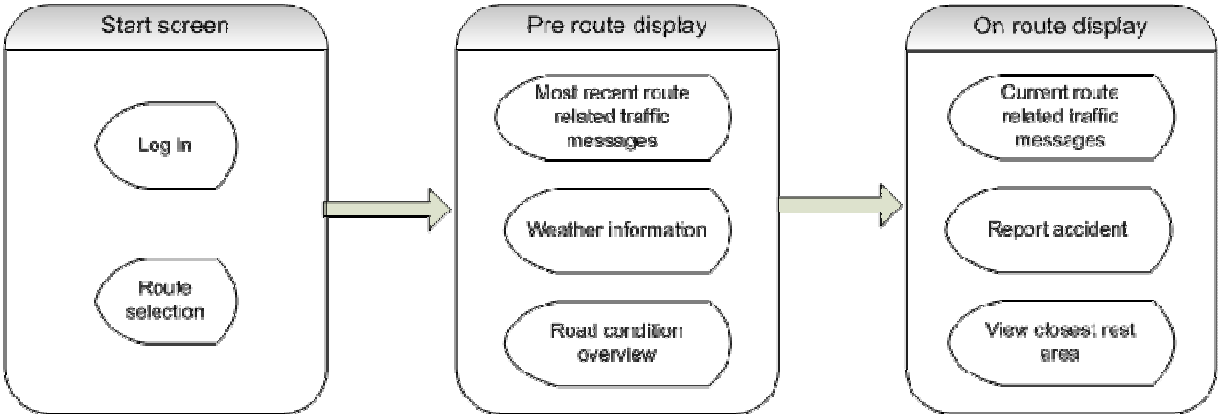


Figure 1. Graphical user interface flow diagram.

H2. The log in display

The log in display connects to the server with the username and password. The users routes are then displayed in a drop down menu and the user can choose which route he intends to travel.

The settings display is intended to be used by the user to enter his username and password so that the user's routes can be downloaded. The settings menu is displayed by pressing the phones menu button.

The drop down menu displays all the users' routes and he can choose the one he intends to travel.

H3. The pre route display

When the user has chosen a route in the previously shown log in display and pressed the OK button the application switches to the pre route display and downloads the most recent traffic messages that concerns the chosen route.

The pre route display shows the user information about which route he has chosen and also the most recent traffic messages affecting that route. The user can also view information about the current weather and download the current road condition overview. When the user feels ready to start his journey he chooses to start the traffic message surveillance by pressing the button at the top of the application.

The user can click on each traffic message and will then receive more information about that message. The user can also choose to view the location of the traffic message on a map.

The traffic message located on a map. The user can again get more details about the message by clicking the flag representing the traffic message.

The user can choose to view the current weather forecast. This forecast is a very detailed forecast for approximately the next ten hours. The forecast is displayed in a list view where the user can scroll and click on the different forecasts.

Each part of the forecast can be clicked to view more details.

The user can also choose to view the current road condition overview. The road condition overview is given by county and gives a general view about the road conditions in the county and can also display closed and damaged roads and other abnormalities.

H4. The on route display

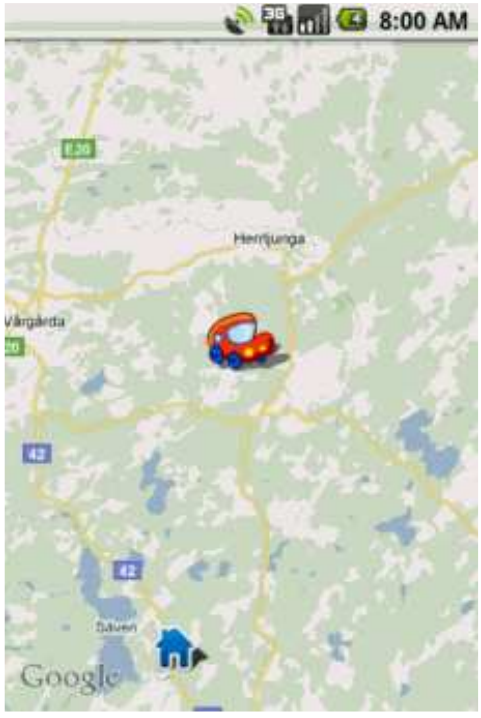
When the user chooses to start his journey in the pre route display he his taken to the on route display. The on route display displays the current traffic messages concerning the chosen route and each time a new message concerning the route is received by the server it is pushed out to the Android application. The messages are presented in a list view, the same way as in the pre route display. The user can also choose to view the nearest rest area and report accidents.

Each traffic message in the list view can be clicked and more details about the message will be displayed.

The user can also choose to view the traffic message on a map as shown in the pre-route display. If the user clicks on the traffic message flag a dialog with more details about the message will be displayed.

If the user sees something that he wants to report he can use the report accident display to do this. A number of different accident types are available and the user can simply click on one of these and a dialog with more details is displayed.

When reporting an accident or other abnormality the user will get a short summary of what the message to be sent contains and the user can choose the severity of the accident before sending the report.



When on route the user can choose to view the closest rest areas. The rest areas are displayed on a map together with an icon showing the users position.

Each rest area icon can be clicked to get more information about that rest area. The information contains availability, information about gas, food, facilities etc.

Appendix I
Communication document

Table of Contents

I1. Introduction

This document will describe the different ways of communication within the system. As seen in figure 1 there is four main communication links to and from the main server. These communication links will be further described in section I2. The communication within the main server will be further described in section I3.

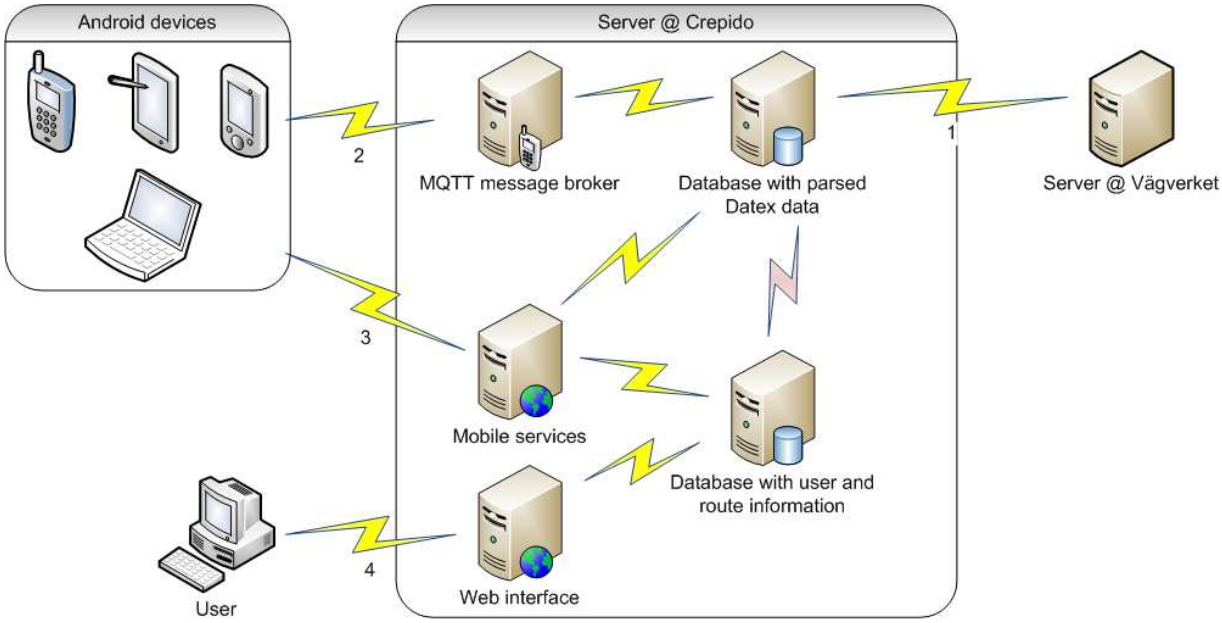


Figure 1. Detailed system overview.

I2. Communication

I2.1 DATEX II

The traffic information that we use in the system comes from Trafikverket. Their data is formatted in the DATEX II format which is a standard designed to make it easier to exchange information between traffic management centers in Europe.

The XML schema for DATEX II will not be displayed in this report because of its size (about 300 pages) but it can be downloaded at;
https://DATEXII.vv.se/wiki/(S(qgnmibvtauwru055ktnqlh45))/GetFile.aspx?File=Dokumentation%2fDATEX IIIISchema_1_0_1_0.zip.

The DATEX II schema makes it possible to transfer a large variety of information and is somewhat complex. When we get a traffic messages to our server we select the most valuable information for us and save it to our database. By doing so we get a much easier way to work with the traffic message data.

I2.2 MQTT

The MQTT message broker is responsible for receiving accident reports from the users and also pushing the traffic messages to the Android application. When a traffic message is received by the server the

server checks which routes that are connected to this message. The MQTT message browser will then push the message to the affected routes. More about the technique behind the MQTT message broker can be read in the report under section 6.2.3 "MQTT (MQ Telemetry Transport)".

When the Android application sends an accident report to the server the server interprets this and stores it in the database. The messages are sent in XML format and the XML format looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<messages>
	<message>
		<userid></userid>
		<accidenttype></accidenttype>
		<latitude></latitude>
		<longitude></longitude>
		<severity></severity>
	</message>
</messages>
```

The traffic messages that are sent to the Android application by the MQTT message broker are also sent in XML format. This XML format is the same as for the traffic message service that can be seen in section I2.3.3.

I2.3 Mobile services

The mobile services purpose is to provide the mobile application with data stored on the main server. The mobile application sends a request to the service, the service gathers the information requested and answers with a XML file.

I2.3.1 User id service

The user id service takes username and password as parameters and returns the users id number. This service is used when a user wants to report an accident (each accident report is sent with the id of the user sending the report).

The request parameters are "user" and "pass". The service checks the username and password and returns an XML file with the user id. The returning XML file can only contain one user id. The XML file looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<userid>
	<id></id>
</userid>
```

If the request was invalid or something else failed during the gathering of data at the server a XML error file is sent. The error file simply looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<error />
```

I2.3.2 User login service

The user login service takes username and password as parameters and returns the user's routes. This service is used for the login on the mobile application. The user needs to validate himself by logging in and in return the user gets all his routes.

The request parameters are "user" and "pass". The service checks the username and password and returns an XML file with the user's routes. There can be several user routes in the returning XML file. The XML file looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<routes>
	<route>
		<id></id>
		<name> </name>
		<description> </description>
		<from_latitude></from_latitude>
		<from_longitude></from_longitude>
		<to_latitude></to_latitude>
		<to_longitude></to_longitude>
	</route>
</routes>
```

If the request was invalid or something else failed during the gathering of data at the server a XML error file is sent. The error file simply looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<error />
```

I2.3.3 Traffic message service

The traffic message service takes the route id as parameter and returns the most recent traffic messages that are within a predefined distance from the current road. The distance variable is defined in the source code together with the date variable that determines the time limit for how old a traffic message can be to be sent with the XML answer.

The request parameter is "routeid". The service then requests traffic messages within the right distance and date from the database and returns an XML file with the traffic messages. There can be several messages in the returning XML file. The XML file looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<messages>
	<message>
		<id> </id>
		<type> </type>
		<comment></comment>
		<lastupdated></lastupdated>
		<locationtext> </locationtext>
		<roadnumber> </roadnumber>
```

```
            <overallstarttime></overallstarttime>
            <overallendtime></overallendtime>
            <longitude></longitude>
            <latitude></latitude>
        </message>
</messages>
```

If the request was invalid or something else failed during the gathering of data at the server a XML error file is sent. The error file simply looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<error />
```

I2.3.4 Road condition overview service

The road condition overview service takes the name of a Swedish county as parameter and returns a road condition overview report for that county. The valid counties are;

- Blekinge län
- Dalarnas län
- Gotlands län
- Gävleborgs län
- Hallands län
- Jämtlands län
- Jönköpings län
- Kalmar län
- Kronobergs län
- Norrbottens län
- Skåne län
- Stockholms län
- Södermanlands län
- Uppsala län
- Värmlands län
- Västerbottens län
- Västernorrlands län
- Västmanlands län
- Västra Götalands län
- Örebro län
- Östergötlands län

The request parameter is "county". The service requests the most recent road condition overview from the database and returns an XML file with the road condition overview. There can only be one overview in the returning XML file. The XML file looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<roadconditionoverview>
        <overview>
            <text> </text>
            <header> </header>
```

```
            <county> </county>
        </overview>
</roadconditionoverview>
```

If the request was invalid or something else failed during the gathering of data at the server a XML error file is sent. The error file simply looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<error />
```

I2.3.5 Queue service

The queue service was designed as a test when exploring other possible mobile applications that could make use of the DATEX II data stored in our database. The queue service takes a road number as parameter and selects the most recent queue messages from that road number. This implementation is very specific to a certain kind of queue messages within the city of Gothenburg.

The request parameter is "roadnumber". Examples of road numbers can be "E 6" or "E 20", the queue service will then select the most recent queue messages from that road and return them as a XML file. There can be several messages in the returning XML file. The XML file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<queueinformation>
        <queue>
            <locationtext> </locationtext>
            <generalpubliccomment> </generalpubliccomment>
            <roadnumber></roadnumber>
            <overallstarttime></overallstarttime>
            <latitude></latitude>
            <longitude></longitude>
        </queue>
</queueinformation>
```

If the request was invalid or something else failed during the gathering of data at the server a XML error file is sent. The error file simply looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<error />
```

I2.3.6 Database service

The database service does not take any parameters and simply returns a database file. The database service is used when the user wants to update the rest area database. The rest area database is rarely updated and is therefore located in the mobile application. If the user wants to update this database it is simply overwritten with a new one from the server.

If the request was invalid or something else failed during the gathering of data at the server a XML error file is sent. The error file simply looks like this;

```
<?xml version="1.0" encoding="UTF-8"?>
<error />
```

I2.4 Web interface

The web interface is where the user manages his routes. The web interface runs on the main server and when the routes are saved to the database all communication between the web interface and the database is done in the source code of the web interface.

I3. Back end

I3.1 Database

The database is the center of the back end. All different data such as user information, route information, user accident reports and DATEX II data is saved to this database. The database then delivers this information on request to the MQTT broker, the mobile services and to the web interface. The database structure can be seen in figure 2. Since this has been a proof-of-concept project we have not put that much effort into optimization of the database. More information about the database can be read in the report at section 6.2.4 "Database for the back end".

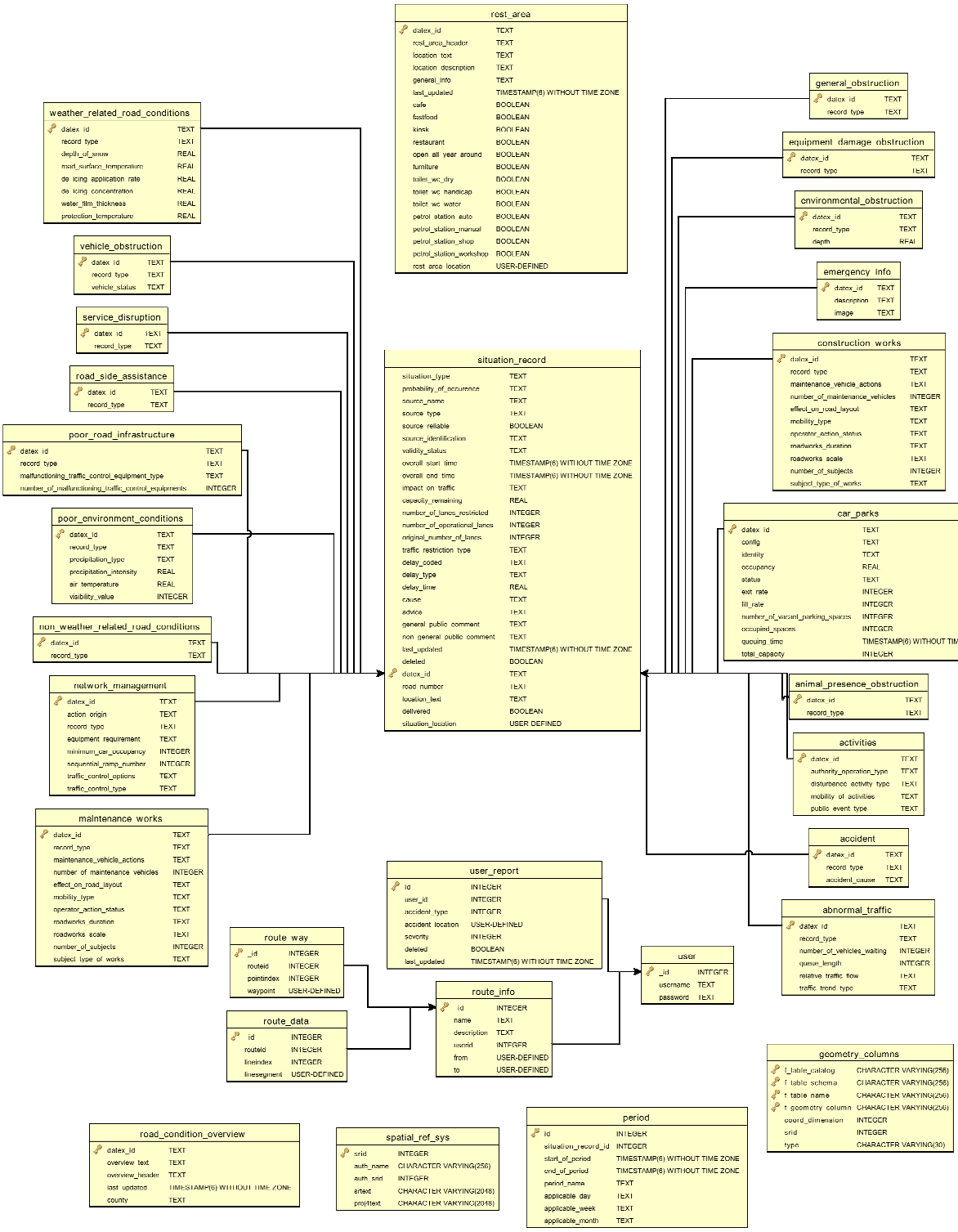


Figure 2 The database structure