A Fleet Management System

Tomas Svensson, Engineering Physics F94
Department of Computer Science,
Lund Institute of Technology,
Lund University, Lund, Sweden
Itinerary Systems AB, Lund

Supervisors:
Jonas Persson, Department of Computer Science
Per Ola Ingvarsson, Itinerary Systems AB

November 30, 1999

Abstract

This thesis describes a fleet management system designed to supervise and control a fleet of vehicles. The system is intended for road carriers or emergency vehicle dispatchers that want to continuously follow the geographical positions of their vehicles e.g. on a map. It allows written communication between drivers and their dispatching central in addition to voice communication, thus reducing the number of misinterpretations and removing the need for spelling using the phonetic alphabet. If a destination is assigned to a vehicle, the program can calculate the estimated time of arrival. The vehicle unit also includes an emergency button, which simplifies rescue operations by providing precise vehicle positioning and route guidance. Customers of a road carrier can be allowed to see where specific vehicles are located, over the Internet, and can automatically be provided with estimated times of arrival, which can be very useful for just-in-time logistical systems.

Contents

Chapter 1

Introduction

This master thesis describes a fleet management system designed to supervise and control a fleet of vehicles and their drivers. The system has been developed to automate and simplify communication for companies with several vehicles on the road. It is also intended to increase the accuracy of, as well as decrease the costs, associated with communication. The system is developed at Itinerary Systems IS AB (ISAB) in Lund, Sweden.

A fleet management system can be used for transferring text messages between a vehicle driver and its central operator. It can be used for vehicle positioning, to follow the progress of the vehicles on a map, and in case there is a known vehicle destination, it can be used to calculate an estimated time of arrival at that destination.

This report is of interest to companies or organizations that consider investing in a fleet management system but feel that they need some more background information. It should also prove to be useful to future developers who want to update the system described here.

1.1 Non Technical Description

Fleet management is the art of supervising and controlling a fleet of vehicles, e.g. trucks or emergency vehicles. The fleet management project at Itinerary Systems enables a central server computer and operator to communicate either with the driver of a vehicle, or with the vehicle unit in that vehicle.

The driver can ask for directions or otherwise communicate with his dispatching central in written form, reducing the need for less accurate voice communication. Voice communication will still be needed, however, but only in more complicated and non-standard situations. The vehicle unit can report the position as given by a GPS receiver as well as other parameters of the vehicle such as speed, heading and airbag status. It can automatically report deviations from a negotiated route or delays as compared to a schedule. This information will enable the central to calculate accurate

estimates of the positions of the vehicles and helps in predicting estimated time of arrival. This is useful, e.g. in just-in-time logistics systems. The positioning information will also help the operator in distributing assignments, e.g. by listing free vehicles close to a cargo pickup point. In addition, the information on position and estimated time of arrival can be made available to selected customers of the dispatching firm via the Internet, to assist them in planning their logistics.

Costs associated with communication, such as telephone charges and time spent talking on the telephone are often an unnecessarily large part of the expenses of dispatch and road carrier firms. A reduction of the time that a telephone connection has to be open, and the time that humans have to be involved in the communication, will help to reduce these costs. Currently the ISAB fleet management system communicates using SMS/GSM, with a network covering most of Europe. Additional benefits of a fleet management system are better vehicle positioning precision and more accurate communication leading to quicker deliveries and a higher utilization of the vehicle fleet.

1.2 Overview of Technical Standards and Definitions

GSM Global System for Mobile Communications, the current standard used for digital mobile radio telephones, mainly in Europe. The GSM service is provided by a network of transmitters and receivers. Each network is run by a company, the *network operator*. The networks of different network operators are linked together so that you can communicate between mobile telephones that are connected to different networks. The telephone number of a mobile telephone in a network is its *network address*. (Mouly and Pautet, 1992) contains more details.

SMS Short Message Service, a standard for sending short text or data messages over the GSM network. The messages are limited to a maximum of 160 7-bit characters, or equivalently 140 8-bit characters. See (Mouly and Pautet, 1992) for a complete description.

GPS Global Positioning System, a system for positioning using a network of 24 satellites, initiated by the US Department of Defense. The satellites continuously transmit their position in space as well as a time stamp. Positioning of a receiver is accomplished by comparing

the different times that are received, from at least three satellites, with the positions stated by these satellites. With this information, it is possible to calculate the position of the receiver on the surface of the earth. If information on at least four satellites is used the altitude of the receiver can also be estimated. The accuracy is in the order of 10 meters. Russia has a similar system of satellites called GLONASS, and the EU is planning one of their own.

SA Selective Availability, is used by the US Department of Defense to reduce the accuracy of positioning using GPS to about 150 meters, by adding a disturbance to the time signals transmitted from the satellites. The US military and some scientific applications get to know the scheme of disturbances and can recreate the true signals, thereby still achieving the better accuracy.

DGPS Differential GPS, is a system for correcting for the disturbances in the signals received from the satellites. The basic idea is to place stationary GPS receivers on positions that are known with a good accuracy. By comparing the known position with the position received from the satellite network, differential corrections can be calculated and transmitted via radio to mobile DGPS enabled receivers. Many countries have set up systems of this type. However, they often demand a license fee for the use of the systems.

RDS Radio Data System, a system for transmitting text information in FM radio channels. The system is used to some extent for transmitting information on traffic conditions in text format. The system can also be used for distributing DGPS corrections.

Coordinates The position of a point on the surface of the Earth can be stated as a coordinate pair: (latitude, longitude). The latitude is the absolute angular difference perpendicular to the equator between the equator and the point, see figure 1.1. If the point is north of the equator the latitude is stated as "*latitude* N", and if the point is south of the equator as "*latitude* S". The longitude is the signed angular difference between the Greenwich observatory in Great Britain and the point, along a line parallel to the equator. Longitudes can be

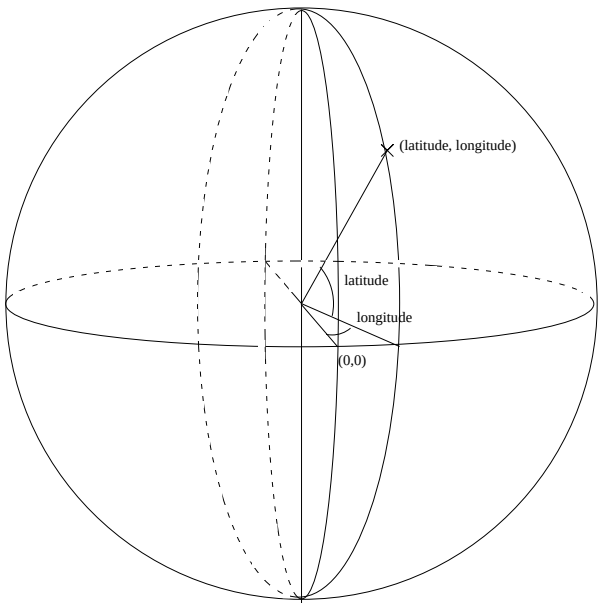

Figure 1.1: Coordinates composed of latitudes and longitudes

stated as "*longitude* E" for points that lie to the east of the zero meridian, and "*longitude* W" (or "-*longitude* E") for points that lie to the west of the zero meridian.

Bitmapped map A map composed of pixels, representing a picture of a map. Bitmapped maps are often scanned from normal paper maps and look similar to paper maps, which can be easier for the average user to interpret. However, there is no easy way for a computer program to tell where the roads are, or to read the names of the locations indicated on the map.

Vectored map A map represented as a graph containing nodes and edges. The edges can e.g. represent roads that may be traversed by a vehicle, while the nodes may represent intersections or traffic rule changes. A vectored map is needed to make route calculations and estimate the driving time and distance between two positions.

1.3 Problem Definition

The problem to be solved in this thesis is to develop a means for reducing the cost of communication for fleet management. The expected result of this

thesis is a program that can be used as a fleet management server in the central computer, and a program to be used as a client in the vehicle units. The software requirement specification is given in chapter 3. The idea was to work according to the following plan:

- Problem formulation. Look for previous work in the same area in academic journals and in publicly available commercial material. Ask potential end users about their ideas.
- Write a software requirement specification and make an initial design.
- Think about how and when communication will take place. Define a communications protocol.
- Work on a preliminary algorithm to estimate the vehicle positions between transmissions.
- Implementation of the server program.
- Design and implementation of a web interface to the server vehicle information.
- Implementation of the client program.
- Simulation and bug fixing.
- Improved algorithm to estimate vehicle positions.
- Implementation and some more bug fixing.
- Write this master thesis.

1.4 Report Outline

Chapter 2 consists of a short summary of requests from potential end users, a description of the functionality currently found in competing systems, and a definition of the problem that is to be solved in this master thesis. A software requirement specification and some performance requirements are found in chapter 3. Chapter 4 describes the surrounding program that the fleet management module will be a part of. Chapter 5 shows the design of the fleet management system. Chapter 6 gives a short overview of the GUI and a few hints on how to use it. Chapter 7 defines the communication protocol. Chapter 8 describes how the position of the vehicle could be estimated. Chapter 9 describes the vehicle unit and its fleet management module. Chapter 10 contains information on the web interface. Indications from simulations and tests are found in Chapter 11. The result of the development and some concluding remarks can be found in Chapter 12.

Chapter 2

Functionality

This chapter is intended to outline what potential end users are interested in, and what functions the competing systems are offering.

2.1 User Analysis

In the beginning of this project there were some ideas of the types of services that would be of interest to a transportation firm utilizing a fleet management system. It was expected to be of interest to know an approximate vehicle position at every point in time for all vehicles in the fleet, and to get an estimate of the time of arrival of each vehicle at its destination.

In order to get a clearer picture of the needs and requirements of potential users of a fleet management system, a meeting with a local dispatch and road carrier firm involved in international transport was set up. The name of the surveyed company will not be disclosed. The firm accepts orders from companies that want large or small volumes of cargo to be transported from one location to another inside Europe. The goods transported vary depending on what the customers request, examples are printing paper and canned food.

From a meeting with only one potential user, it is of course impossible to get a complete overview of what information could be useful to most users of fleet management systems. All the same, since the author of this thesis had very little experience from the field of transportation he found the visit very constructive and worthwhile.

The company currently used voice communication over mobile GSM telephones to handle all communication between the central office and the vehicle drivers. The office used the telephones to tell the drivers when and where to load or unload cargo, and to ask for their current position, predicted times of arrival and general status. The drivers used their telephones to ask for directions, to report the success of their tasks and to some extent to communicate with their colleagues in other vehicles. A lot of time reportedly was

wasted on the telephone trying to get and understand driving directions in foreign languages, especially when having to spell using a foreign phonetic alphabet.

The average vehicle driver did not have much computer training, and extra care should be taken to make the vehicle unit easy to use and more or less self-explanatory. The vehicle unit must be kept very simple, and it must be possible to switch off any potentially annoying functions. Some drivers can be expected to have a hard time reading without glasses so the printed text and the graphical text display should be large, as should the keys. The keys of a mobile telephone were indicated as the lower limit in size.

The more experienced drivers should be able to find their way on their own, and would not want an electronic device to try to tell them a better way. The basic principle initially should be not to interfere too much with the driver and his work, unless it is reasonably certain that he can be helped by the system.

Also, there is always a chance that a calculated route from a digital map is not the optimal one for a large truck. In fact, the driver might be expected to be upset should he be led into a narrow passage by the vehicle unit. Especially if the road in reality turns out to be blocked and the driver has to back up out of the street with his trailers. The risk of the driver losing his temper in that kind of a situation would be much smaller if the driver had chosen that route himself. The company representative saw several potential problems with the map information available for route guidance. For example, some roads and bridges put an upper limit on the weight of the passing vehicles, which has to be taken into account when calculating a route. Currently the roads limiting the total weight of this company's vehicles are in Austria with a total weight limit of 38 tons. The height and turn radius of the vehicles also limits their access to narrow streets enclosed by multistory buildings with projecting balconies. These problems can be overcome by using sufficiently detailed digital maps. Route guidance is outside the scope of this thesis.

As it turned out, the dispatching company was quite informed of the currently available systems, and they knew for what purpose they wanted to use them. First, they want to be able to know the location of their vehicles in their dispatching central, and preferably they wanted the locations to be presented on a map. They would use this function e.g. if one of their customers called and asked them if they could carry some cargo from one specific point to another. The system would help them check if they had a vehicle close to the point of origin, and help them to determine a fair price, which in part was determined by the driving distance.

The dispatching operator want to be able to get accurate status reports immediately after successful deliveries in order to write an invoice shortly afterwards. Their current procedure is to wait for the driver to return with the papers to the office, adding several days to the customer credit period

and the cost cycle, and thus leading to an increased cost of interest.

The dispatching company wants to be able to present the positions of some of the vehicles to their customers over the Internet, preferably using the World Wide Web. In doing so, it is important not to give away too much information, and thus the vehicles should disappear from a customer's map (or other form of presentation) once her order is completed. Any vehicles not currently used by a customer should be invisible to her. One of the reasons for not wanting to disclose the position of all vehicles to every customer is that quoted prices could be put in question. If a customer e.g. could see that the dispatching company has a truck right next to her, she could demand a discount.

The function to optimize the matching of vehicles and cargoes offered by some fleet management systems is not interesting. This is because the current stage of development of the optimization routines are believed not to take all relevant information into account in a better way than if a human performed the optimization.

They were not quite satisfied with their current system using voice communication over mobile phones, which caused very high phone bills and a lot of routine work for both driver and the central operator. An attempt had been made to reduce the frequent misinterpretations of a voice connection by installing fax machines in the vehicles. This was not a success, and those machines were later removed. By introducing a computerized fleet management system, they hoped to be able to reduce their telephone bills to less than half of its current amount.

2.2 Competing Systems

There are a number of competing systems, offering more or less sophisticated and useful functions. Prospectuses from a number of competing companies have been studied, such as (folder from Blaupunkt, 1998), (folder from PTV, 1998), (folder from SkyCom, 1998), (folder from Cellpoint, 1998), (folder from Knud Hansen, 1998), (folder from TeleTec, 1996), (folder from Volvo, 1999) and (folder from Passo Fleet, 1998). The actual use in the transportation industry of these kinds of systems is currently very small, but there are generally high expectations on the future growth of the market for fleet management systems.

There are systems that offer the ability to monitor some states of the vehicle and its load, and to report any deviations from the usual states. Some systems make it possible to report the state of the ignition switch, e.g. to assist in making a drivers log. If the airbag trigger is set off this could be reported and initiate a distress call. A system used in South Africa, see (Commercial article from G-Track, 1997), makes it possible to report if the cargo doors are opened or the trailer is hooked off at an unauthorized posi-

tion, thereby helping to reduce the large losses of cargo in troubled regions, such as parts of South Africa. For refrigerator trucks and animal transports, reports can be made when the temperature deviates from a normal interval. This kind of monitoring of states is beyond the scope of the current system as described in this thesis, although nothing prevents future extensions of the system and the addition of the necessary external sensors. As long as there are free external ports in the vehicle unit, and if the necessary extra programming in the server and vehicle unit can be easily performed, the addition of these external sensors could be performed in a relatively short time. Care should be taken to design the system so that extra sensors can be added easily and that the software can be updated both in the vehicle units and in the server.

Most systems let the driver report on their current activity, e.g. by pressing a key marked *Loading, Unloading, Break, Fueling, Service, Rest, Distress, Traffic jam* or *Under way*. This requires the active participation of the driver to press the appropriate key once this state changes. A few systems try to automate this process as much as possible by declaring default states when certain conditions are met, e.g. when the driver has stopped at a destination with a predetermined activity such as *Loading* or *Unloading*.

One popular function is the ability to initiate a distress call with the press of a button. Being quick and easy to use are important characteristics of such an alarm button. The button could be utilized to notify the police if the driver is assaulted, or to contact the nearest vehicle repair center if the vehicle breaks down. Being able to give precise positions or even address and driving instructions to the rescue service center simplifies the rescue operation considerably.

Practically all systems have functions to follow the positions of the vehicles on a map, with the positions being updated on regular time intervals. The information is often presented on a road map in the form of a bitmap. The journey can be reviewed and the driven route is documented. The information that can normally be saved is position, speed, heading and a time stamp. A few systems offer calculation of estimated time of arrival, thus requiring either a vectored map or a human operator capable of interpreting a bitmapped map. Estimated times of arrival are important information in *just-in-time* logistics systems. To perform the calculations the destination must be known and entered, e.g. as *city name, address, company name* or *customer id*.

A few systems offer the ability to define an approved route that the vehicle is expected to follow, and if the vehicle deviates from this route, either in time or space, the operator is notified. This is used, e.g., in the transportation of radioactive material or valuable goods (H. Sannen, 1998).

Messages can often be transferred to the vehicle unit in text format, even when the driver is not in the vehicle. Messages of new and carried out assignments can thus easily be transferred. Some systems even send read

confirmations back to the central after the driver has viewed the message for a predetermined time. Sent and received messages are logged in the server, and to some extent in the vehicle units.

Techniques for applying differential corrections to the GPS/SA signals in the server are described in (Papadoglou, 1998), whereby a positioning precision of down to 10 meters can be achieved. This has the added benefit of reducing the cost of hardware in the vehicles as compared to ordinary DGPS receivers, in that they do not require a receiver for differential corrections in every vehicle. One might also suspect or even assume that the transmission of these corrections are not standardized all over Europe, thus requiring the capability to receive differential corrections from several different systems, increasing the cost of in-vehicle differential corrections. By placing only plain GPS receivers in the vehicles, the overall hardware cost will be lower. If the information received by these plain GPS receivers can be sent to the central, where the differential corrections are applied, the probability that the corrections will work is higher.

A few systems claim to take variations in traffic conditions into account when making route calculations. Information on traffic conditions is most often taken from transmissions over the RDS (Radio Data System) FM radio network by the vehicle receivers.

To sum up, this chapter has outlined user requirements and existing systems. The users require a system for communication and positioning. They want to get status reports after finished assignments. They also want to provide a web interface with information for their customers. The functions and capabilities of existing systems have been indicated.

Chapter 3

Software Requirement Specification of the Fleet Management System

This chapter describes the requirements on the final system. The requirement specification originates in the user survey and to some extent in ISAB tactical planning.

3.1 Definitions and Terms

1. A *fleet* is a set of vehicles under a common command.

2. A *vehicle unit* is a small computer in a vehicle capable of communicating with a central server.

3. A *fleet management system* is a system for supervision and control of a vehicle fleet. Most fleet management systems consist of a central server computer, a vehicle unit in each vehicle, and a means of communication. The system can be used by, e.g., a road carrier to follow the progress of its vehicles. A human operator monitors the system and communicates with the drivers.

4. A *standardized message* is a message with a content that is largely known to the recipient. Consider the following example message: "I have just finished loading A units of B at C. Next, I'll be going to D.". This kind of message is probably sent more than once, with the letters A, B, C and D replaced with different strings of text. If the template for this message is stored in both vehicle unit and server, they can communicate by sending the message template number and then list the strings of text to insert into the blanks of the template. This way of reducing the size of a message utilizes the fact that people

often send the same kind of messages several times, with only slightly different content. The parts that are different or changed are filled in by the sender, and are the only ones that are sent to the recipient of the message. The recipient will have to decode the message into a human readable form by merging the template text with the received strings of text.

3.2 Goal

The aim of this project is to implement a central system and a vehicle unit program. Both of them should be easy to use. The central system will be implemented in Delphi, an object oriented visual programming language similar to Pascal, see (Cantù, 1998). Delphi is used to quickly develop programs with graphical user interfaces. Also, the previous system MapCentral, see chapter 4, was written in Delphi. The vehicle unit program will be implemented in C, whereas the web interface will be implemented in C++, see (Stroustrup, 1997) and (Lippman and Lajoie, 1998). C is used for the vehicle unit program simply because we have a C compiler that can compile C programs for the vehicle unit computer. C++ is used for the web interface because our new web-server is written in C++.

The central system being able to:

1. Send a free text message to a specific vehicle.
2. Receive a free text message from a vehicle unit or a driver.
3. Send a free text message to all vehicles.
4. Send a standardized text message to a specific vehicle or to all vehicles.
5. Receive a standardized text message from a vehicle or driver.
6. Initiate a voice connection to a specific vehicle on the operator's request.
7. Select a vehicle, and send orders to the driver or vehicle unit.
8. Follow the positions of the vehicles as symbols on a map.
9. Select a vehicle by pointing to its symbol on the map.
10. Follow the vehicles in a table listing coordinates, activities and other information.
11. Select a vehicle from the table mentioned above.

12. Extrapolate the position of a vehicle on the map to give an approximate position also between its position reports.
13. Change the position report intervals of the vehicle unit.
14. Display the driving history of, and messages sent to and received from a vehicle.
15. Show information over the Internet using a web interface, about the vehicles engaged with a certain customer.
16. Receive distress calls from vehicles and display information to the operator.
17. Create and edit vehicles.
18. Fleet and vehicle information can be written to and read from disk.

The vehicle unit being able to:

1. Send a free text message to the central server.
2. Receive a free text message from the central server.
3. Send a standardized message to the central server.
4. Receive a standardized message from the central server.
5. Report its position to the central at times and/or positions given by a specified schedule.
6. List the received messages.
7. Display a message to the driver.
8. Initiate a voice connection to the central operator on the vehicle driver's request.
9. Initiate a distress call to the central.

3.3 Performance Requirements

The response time to user input is not critical, with an acceptable delay of up to a few seconds in the worst case. The system must however be able to handle up to several hundred vehicles at the same time, thus making it suitable for small and medium sized dispatching companies.

3.4 Use Case

A road carrier operator gets a call from one of his customers. The customer has some goods to be carried from a position a and wants a quotation of a transport of the goods from a to b. The price of the road carrier is in part determined by the distance that a vehicle must drive to collect the goods. In looking at his computer, he selects the vehicle that appears to be closest to the goods and that he knows has enough free space. By requesting a position report from that vehicle, he will (hopefully) in less than a minute know the approximate position of the vehicle, without having interfered with the driver. The operator can quickly estimate the distance by looking at the map, or in case he is using a vectored map, he can let the program calculate the distance. The road carrier will then be able to quote a fairer price than would be possible without the system. The customer recognizes the price as being fair and accepts the offer. The operator then enters the order as a standardized message, by filling in what to carry, where to collect it and where it is to be delivered and then sends the message to the selected vehicle. The driver reads and accepts the order. Should he later forget some part of the order he can always let the vehicle computer recall it for him.

To sum up, this chapter has presented a list of requirements of the final system, both on functionality and on performance.

Chapter 4

Program Environment

This section describes relevant parts of the main program that the server side fleet management server module will have access to.

The fleet management system is an add-on module to the ISAB program MapCentral described in this chapter. The MapCentral program is mainly used for finding locations and generating optimal routes between points on a map. The program is currently under development at ISAB, but is fully usable in its current state. A few add-on modules other than the fleet management module have been developed, but they are not used in this thesis and their purpose will remain undisclosed for the time being.

The program can display a vectored map of a specific area, including roads, parks, water and buildings. It is possible to zoom into or out of a region on the map. It can draw street names and indicate traffic rules next to the streets. It can generate and draw optimal routes between an origin and one or several destinations. MapCentral can also give route guidance, among other things. The information used to draw the maps is also useful when generating route descriptions. One can insert delays in the streets to simulate heavy traffic and let the program regenerate the route to see if there is a better route avoiding the heavy traffic.

There is an interface for communicating with cellular telephones, specifically using the SMS protocol. It is thus possible to send a string of data to a particular telephone or vehicle given its telephone number. If the telephone is attached to a vehicle unit, the server can communicate with the vehicle unit. MapCentral has a database of all the valid vehicles, their telephone numbers and some other information.

If a vehicle sends a message in the form of an SMS message to the central, it is received by the MapCentral program. If the message is originated by a valid originating address, MapCentral distributes the message to the intended recipient module, such as the fleet management module. To identify the intended recipient one could look at the originating telephone number and see which services that number subscribes to. However, this would not

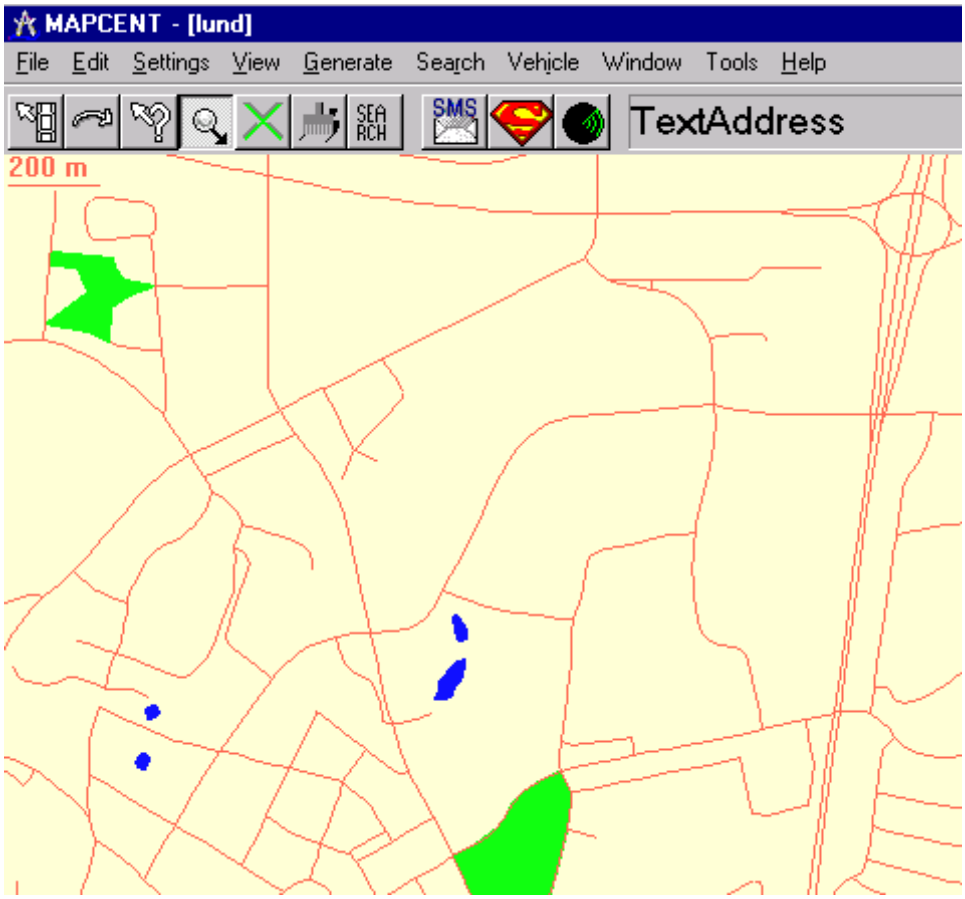

Figure 4.1: The MapCentral main window

be possible if one telephone number subscribes to several services. This could be solved by having separate server side telephone numbers, each providing its own unique service. Another possibility, which is used here, is the standard way of adding a header identifying the type of the message and thus the intended recipient, see chapter 5.

There is also an interface to a web server for providing various types of dynamic information over the Internet. The web server also handles such issues as login, user identification and encryption of the transmission that is necessary to be able to provide sensitive information over the Internet. MapCentral has graphical routines for displaying objects, e.g. symbols representing trucks, on the map. The web server can generate graphical maps of a specific region and draw some additional information such as a route or a vehicle, on those maps.

MapCentral also has routines for associating a position with a name and vice versa. This enables the user to search for a destination by street or company name instead of having to enter the coordinates of the destinations, in the format (latitude, longitude).

MapCentral can convert a coordinate to strings that more readily describe the position using country, city and street names. Another simple, yet usable function converts a heading, e.g. in degrees to a point of compass to simplify the interpretation for the user.

Figure 4.1 is taken from the MapCentral main window. It shows a map of some parts of Lund University. We can see two small lakes close to the Department of Computer Science. Also easily recognized is the highway E22 in the east and the Tuna park in the southern part of the map.

This chapter has outlined some of the functions of the main system that can be used by the fleet management server module.

Chapter 5

Design

This chapter contains an overview of the design of the fleet management system. It will outline the major classes and objects, their interdependence, and their respective responsibilities.

The fleet management system consists of two different computer systems, as shown in figure 5.1. The server part is designed to run on a standard desktop computer, while the vehicle unit programs run in the vehicle computers. A fleet management system implemented using the current design consists of exactly one server and zero or more vehicle units.

5.1 Fleet Management Server Design

The fleet management server is shown in figure 5.2. It consists of the GUI (Graphical User Interface), a communications device and a representation of the fleet. The server communicates with the vehicles and with the users.

5.1.1 Fleet

The *fleet* consists of a number of *vehicles* and a number of vehicle groups, as shown in figure 5.3. It is possible to add and delete *vehicles* or *vehicle groups* to the *fleet*, and to save and load the data on disk. A *vehicle group* consists of zero or more *vehicles* and a *vehicle* can belong to zero or more *vehicle groups*. A *vehicle group* is used to group *vehicles* together that are logically

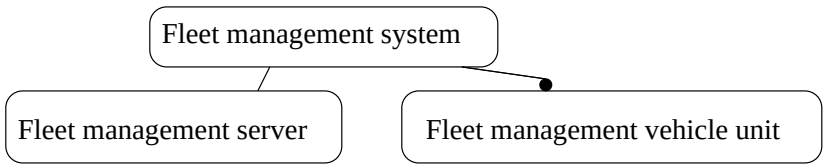

Figure 5.1: System overview

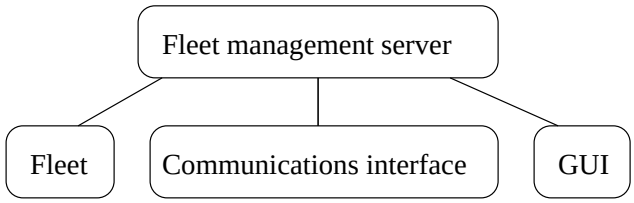

Figure 5.2: Fleet management server

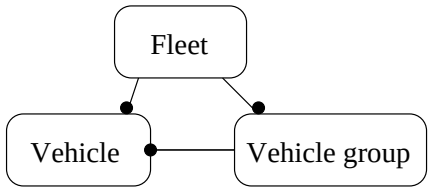

Figure 5.3: The fleet

similar in some respect, such as by the load they can carry. This grouping of vehicles is intended as an aid to the user in organizing the vehicles, and is presently not needed or used by the server itself.

5.1.2 Vehicle

The server side *vehicle* can be interpreted as a shadow representation of relevant parts of the real physical vehicle. The vehicle object, shown in figure 5.4, holds some basic information about the vehicle in the form of attributes which are not shown in the figure, such as its maximum load, the phone number of the embedded system, and the registration plate text. It also has lists of messages sent from the server to the vehicle and messages sent from the vehicle to the server, as well as lists of any performed and scheduled tasks. The *tracker* of the vehicle object keeps track of where the vehicle was located at the time of the last transmission, and can calculate estimates of where the vehicle is currently located. The *message handler* converts and sends messages to the vehicle, and receives and interprets messages from the vehicle to the server. The *message handler* uses the *communications interface* of the *fleet management server* to communicate with the vehicle unit.

5.1.3 Design of the Server GUI

This section on the design of the GUI will indicate what views can be displayed and how to move between them from the users' point of view. Figure 5.5 shows the major views. The *main window* lists information about the

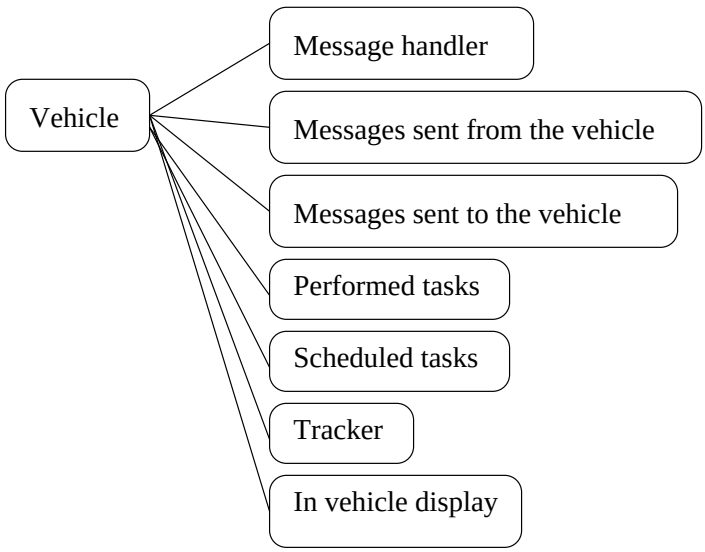

Figure 5.4: The vehicle object in the server

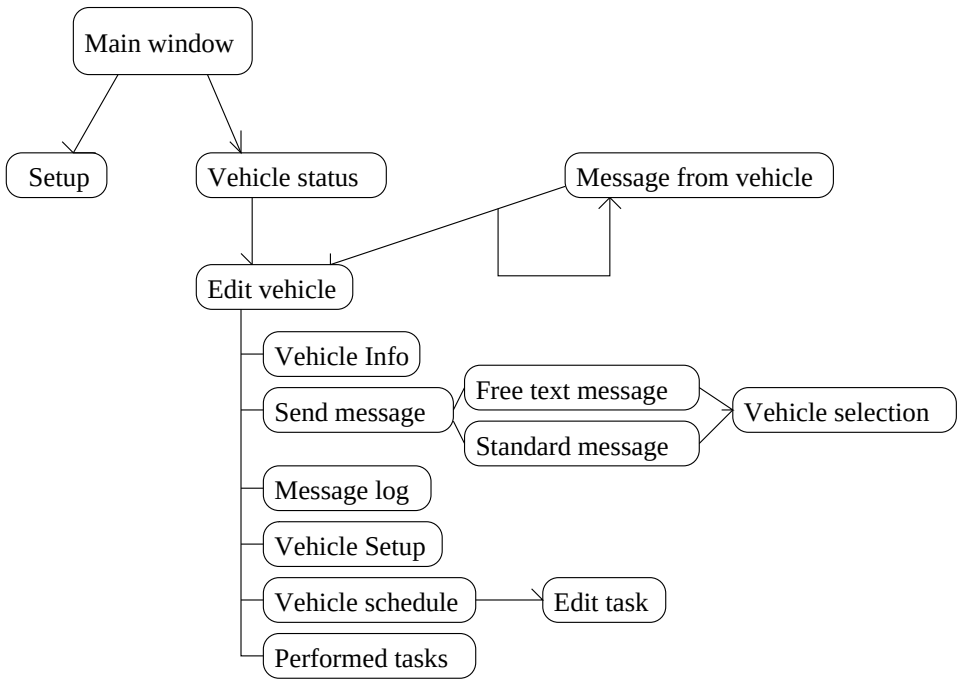

Figure 5.5: The GUI from the perspective of the fleet operator

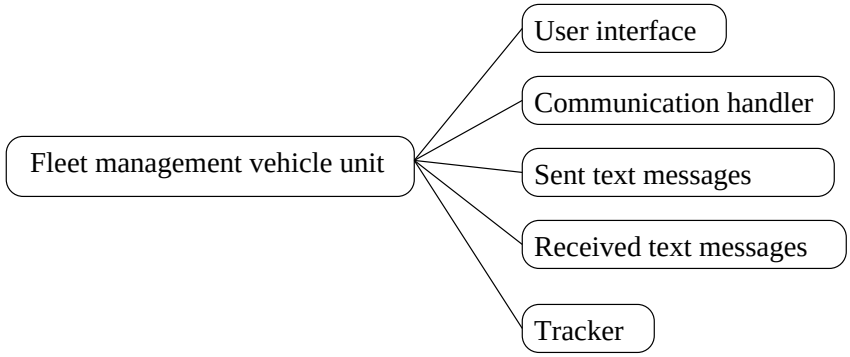

Figure 5.6: Fleet management vehicle unit

fleet, such as the total number of vehicles and the number of vehicles in use, and also allows the user to manage the vehicle groups. From the main window the user can go to the program *setup* to set preferences and various server information, or to the *vehicle status* window. The *vehicle status* window lists the vehicles in a vehicle group or all the vehicles of the fleet, giving some limited information about each vehicle, such as their estimated location and their destination. Here the user can add and remove vehicles. If a vehicle is selected from the list the *edit vehicle* window pops up. From the edit vehicle window, the user can select what she wishes to do with the vehicle. It is possible to view detailed information about the vehicle's current location and status, send different kinds of messages, view the message log, change vehicle settings, edit the vehicle schedule, and view the performed tasks log.

If a text message is received from the vehicle the *message from vehicle* dialog pops up, showing information about the vehicle at the time the message was sent, and the message text. Here the user has the options to open the *edit vehicle* dialog on that vehicle, simply close the window, or if there are more messages waiting to go to the next message. More on how to use the GUI and some screenshots can be found in chapter 6.

5.2 Fleet Management Vehicle Unit Design

The program in the embedded system, as shown in figure 5.6, has a user interface displaying menus, messages and other information to the user, a communications handler for sending and receiving messages, lists of sent and received messages, and a tracking object. The tracking object keeps track of the position of the vehicle and determines when it is appropriate to make a position report to the server.

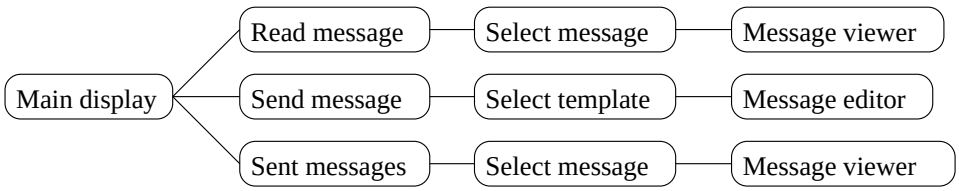

Figure 5.7: The menus of the vehicle unit user interface

5.2.1 Vehicle Unit User Interface

The user interface is organized in menus according to figure 5.7. The default display is the main window, showing the current time and date, vehicle speed and heading, GPS coordinates, and the number of new and unread messages.

If the menu button is pressed, the user can step through the menus. The first choice lets the user list the incoming messages and select one of them. If a message is selected its contents is displayed. The next choice lets the user send a standardized message. The user has to select the type of message template to use, and can then fill in the blanks of that message. A particular type of message template is the free text template that lets the user enter all the text by himself. The fourth selection lets the user review what text messages he has sent to the central server.

If the panic button is pressed, not shown in the figure, the vehicle unit immediately transmits the position and some additional information to the server. The vehicle unit also attempts to initiate a voice connection, independent of the current state of the user interface.

5.2.2 Tracker

The *tracker* keeps track of the position of the vehicle. It determines when it is time to transmit information to the server given the current position report setting. If the vehicle sensors have a new measurement, the *tracker* collects that data and determines if the position of the vehicle is too far away from the route, if the vehicle is lagging behind its time schedule, or simply if a specified time period has passed since the last position report. If this is the case, the *tracker* assembles a message of the current state of the vehicle, its position and optionally some information on the route that was followed by the vehicle to arrive at the current position. It also updates some internal variables that can be used by other parts of the vehicle unit program to determine the most recent and accurate position estimate.

To sum up, this chapter has described the design of the fleet management server program, the fleet management client program and the GUI of the fleet management server as well as the UI of the fleet management client.

Chapter 6

User Interface

6.1 Description of the GUI

The graphical user interface is implemented using the standard visual components of Delphi under Windows. The aim has been to make the user interface to function similar to the user interfaces of other Windows based programs, thereby reducing the learning time for users familiar to standard Windows programs. The aim has also been to make the user interface intuitive and to use clear labels, so that the user will know what to do and what to expect if she issues a command. Section 5.1.3 indicates what windows or forms have been defined, and how they are interrelated. The rest of this section describes some selected forms in more detail.

Figure 6.1 is a screenshot of the *vehicle status* window. It lists all the vehicles of the fleet or of a selected vehicle group. The information in the window is automatically updated by the program as time passes and when new information becomes available. However, the user can force an update by pressing the update button. By a press of the close button, the window is closed. This is the default action when the user presses the *enter* key. If a vehicle is selected when the *delete vehicle* button is pressed, a confirmation request pops up, and if the reply to this request is positive the selected vehicle is deleted from the fleet and any vehicle groups it belonged to. This delete function could be used on the few occasions that a vehicle is sold or the use of a vehicle is discontinued. The drop-down selection list lets the user select a different group of vehicles to list in the window.

The *always on top* check-box lets the user change the behavior of the window. If it is checked, the window will stay on top of other windows, even when the user clicks in the other windows. This is the default behavior of the program and probably most often the desirable behavior. If it is not checked and the user clicks in another window the *vehicle status* window is moved below that other window and can thus disappear from view.

The next line in the same window informs the user about the number of

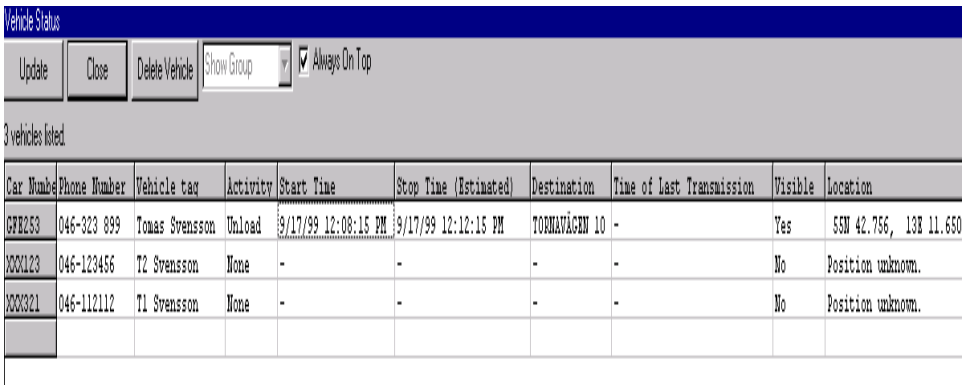

Figure 6.1: An overview of the fleet vehicles

vehicles that are in the vehicle group being listed. The table starts below, the first line of the table being the header and each successive line representing a vehicle. The first column of the table lists the registration number of each vehicle, the second column the telephone number of the vehicle. The next column lists the vehicle tag, i.e. some string of text that can be used to more easily identify a vehicle than the registration number. If each driver has his own vehicle this might be equal to the driver name, but it is up to the server operator to assign vehicle tags. The *activity* column lists the next activity of the vehicle, i.e. what will happen once the vehicle arrives at its destination. The *activity* is consequently not, as one might be led to believe from the column name in the header, the current activity of the vehicle. The *start time* column gives the starting date and time of the vehicles current activity. The *stop time* column gives the estimated finishing date and time of the vehicles current activity. The *destination* column shows the destination of the vehicle, or "Unknown" if the destination is unknown. The *time of last transmission* column shows the date and time of the last message that was sent from the vehicle, or "-" if no messages have been sent from the vehicle yet. The column *visible* sets whether or not the vehicle should be drawn on the map. The *location* column gives the estimated position of the vehicle as a pair of coordinates.

If the user double clicks on a vehicle row in the table of figure 6.1, the *edit vehicle* window pops up, see figure 6.2. The *edit vehicle* window is a tab-sheet, each tab letting the user work with the selected vehicle in a different way. The *vehicle info* tab shown in the figure gives detailed vehicle information, and is intended as the main vehicle information window. The field labeled *current activity* on this sheet is in fact, as previously noted, not necessarily the current activity of the vehicle, but the activity that will commence once the vehicle reaches its destination. The next field, labeled *destination* shows the destination of the vehicle.

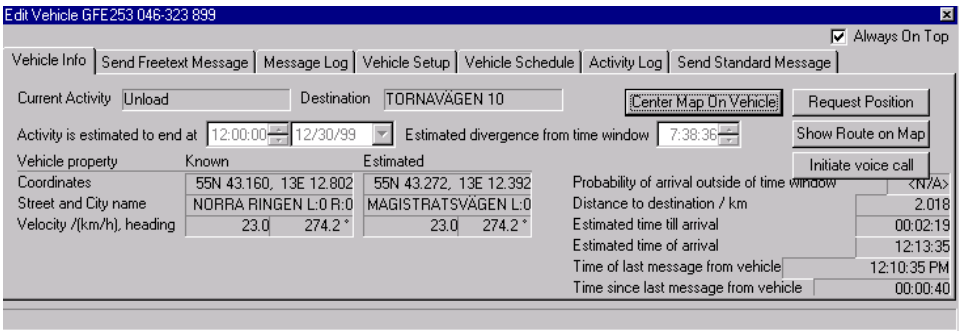

Figure 6.2: Some detailed information on one of the vehicles

If the user presses the *center map on vehicle* button, the map window of the MapCentral program will be centered around the vehicle and an appropriate zoom level will be selected. If the user presses the *request position* button, a position request will be sent to the vehicle, i.e., a report position order of type 0 will be sent to the vehicle, see chapter 7.2. If the button *show route on map* is pressed, the route traversed by the vehicle will be highlighted on the map, and if there is a known destination, the planned route to that destination will be highlighted. A press of the *initiate voice call* button should attempt to initiate a voice connection to the driver of the vehicle. The time and date fields labeled *activity is estimated to end at* shows the time and date that the activity is estimated to end. If there is a time window associated with the activity, i.e., if the vehicle is required to arrive at its destination in a certain time interval, the next time-field, labeled *estimated divergence from time window*, shows by how much the vehicle is estimated to deviate from this time window.

The bottom left table of figure 6.2 gives the position of the vehicle in different ways, as well as its speed and heading. The *known* column contains information on where the vehicle was last known to be located, and its last known speed and heading. The *estimated* column contains information on where the vehicle is currently estimated to be located, and its currently estimated speed and heading. The different ways that each position is stated are by coordinates, i.e. latitude and longitude, and street and city name, i.e. the street and city associated with the coordinates (unfortunately the city name did not fit into the box in figure 6.2). Next to the street name are the street numbers on the left ($L : 0$) and on the right ($R : 0$) hand sides of the street. The table on the bottom right shows how far the vehicle is from the destination, its estimated time of arrival, and its estimated time until arrival. It also shows the time when a message was last sent from the vehicle, and how much time has passed since the last message was sent from the vehicle. The field labeled *probability of arrival outside of time window* is

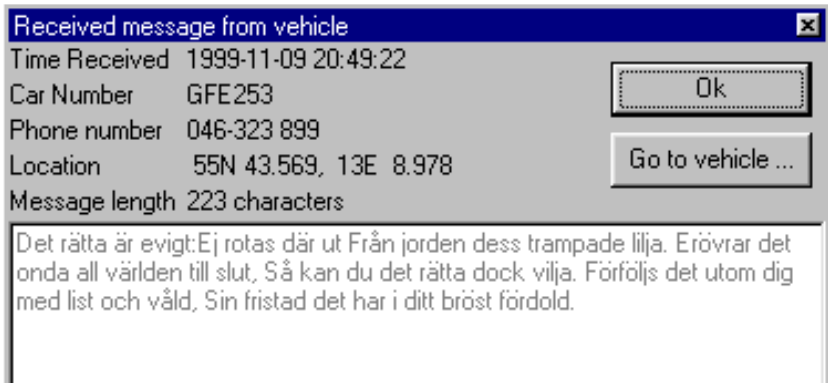

Figure 6.3: An example message from a vehicle. The driver is citing (Tegnér, 1810).

intended to show what the probability is that the vehicle will miss its time window. However, this calculation has not yet been implemented.

The next tab, *send free-text message* is used to enter a free text message and send it to the vehicle. Here the user can also select any additional vehicles to which the message is to be sent. The tab labeled *message log* is used to view the log of sent and received messages. The *vehicle setup* tab is used if you want to change some vehicle properties or the position reporting mode. The vehicle properties that may be reviewed and changed are, e.g., the vehicle tag, the registration number, the telephone number, the size of the shadow vehicle display, and the maximum carrying capacity. The *vehicle schedule* tab lets the operator review and edit the scheduled events, whereas the *activity log* shows a log of the tasks that have successfully been carried out. The tab labeled *send standard message* lets the user select a message template, fill in the blanks, and then transfer the message to the vehicle. The *send standard message* tab could have been placed closer to, or even merged with the *send free text message* tab.

Figure 6.3 depicts the window that pops up when a vehicle sends a text message to the server operator. The top half lists the time that the message was received, the registration number, the telephone number, the location of the vehicle stated as coordinates and the length of the message measured in number of characters. The message in figure 6.3 is 223 8-bit character long, and was split in two 140-byte or less SMS messages including the headers. If the coordinates cannot be readily interpreted by the operator, which is probably the normal case, she has to press the *Go to vehicle ...* button to bring up the *edit vehicle* window. If another text message arrives while she still has the first message window open, the *next message ...* button will be enabled. The text in figure 6.3 entered by the simulated driver is an excerpt from (Tegnér, 1810), and illustrates that message contents do not necessarily

have to be commercial in nature.

This chapter has described some selected parts of the fleet management server graphical user interface to give an indication of how the operator's interface can be used.

Chapter 7

Communication Protocol

This chapter will describe the communication channel between server and vehicle units, and define the communications protocol.

7.1 Communication Channel

Most of the communication between server and vehicle units is performed using SMS/GSM (The Short Message Service of the Global System for Mobile Communications).

This channel of communication has a few limitations.

- There is a cost associated with the transmission of each SMS message. The cost is about $0.25 per message for small volume users (see e.g. the price-list of the Swedish network operator Europolitan of November 12, 1999), while large volume users can negotiate prices in the neighborhood of $0.05 per message.

- There is a limit of 160 7-bit characters per message. To make the programming task simpler short routines for converting 140 8-bit characters to 160 7-bit characters and vice versa have been written. These routines are called just before a message is sent, and just after it is received. In that way, all internal computations can be made without the 7-bit restriction. The 140-character limitation forces the communication protocol to be compact, and the amount of transferred data to be small. If more than 140 bytes are to be transferred at the same time, the data has to be split in smaller messages, each 140 bytes or less in size.

- There is a transmission time of approximately 10-30 seconds or more between the time the originating telephone starts sending the message to the network operator and the time the receiver has received the message from her network operator. There is however, no guarantee

whatsoever as to the transmission time and in some cases it might take several hours before the message is delivered. The messages are automatically time stamped by the receiving network operator.

- One cannot be certain that a sent message is received by the recipient. Therefore, if a message is really important the recipient should be required to reply with an acknowledgement, and if there is no such reply within a certain time the sender resends the message, stating that it is a re-send.
- Some telephone models, even those from well known telecommunication companies, that could have been used in small scale fleet management systems have difficulties receiving and transmitting data at the same time, causing a possibility for data loss at times of heavy communication traffic. These telephones should be avoided if possible.

7.2 Definition of Protocol

The communication protocol used for communication between the server and the vehicle unit is defined below. The communication protocol has been designed to be compact, so that most messages will easily fit into one SMS message. There is still plenty of room for future extensions to the protocol. All communication takes the form of a *message* being sent from the server to a vehicle or from a vehicle to the server.

The format of the definition is BNF, which is a standardized language that can be used to define e.g. protocols and data structures. The following list gives some useful hints on reading the BNF format for those not familiar to it:

< **identifier** >	denotes an identifier.
::=	reads "is defined as".
{ identifier }	denotes zero or more of the specified identifier.
|	logical or.
x:	reads "if the value of the identifier is x then".
//	the rest of the line is a comment, unless the double slash is inside double quotes.

Message

```
<MESSAGE> ::=
   <MESSAGE IDENTIFIER>
   {MESSAGE PART}                    // at least one message part
```

A *message* can consist of one or several *message parts*, thus allowing different kinds of information to be transferred in the same SMS message, such as text and position information.

Message Identifier

```
<MESSAGE IDENTIFIER> ::=
   "//FMM" (5 byte string) // FMM is an abbreviation of
                           // Fleet Management Module.
```

The message identifier identifies the message as originating from a fleet management unit. All valid messages must begin with this identifier. In addition to requiring that messages begin with this identifier the receiving party will also ignore messages with an incorrect originating address, i.e., an incorrect telephone number.

Message Part

```
<MESSAGE PART> ::=
   <MESSAGE TYPE (1 byte)>
   <MESSAGE BODY>
```

The definition of the message body depends on the value of the message type.

Message Type

```
<MESSAGE TYPE> ::=
   <0: free text> |
   <1: standardized message> |
   <2: voice call initiation> |
   <3: report position order
       (only sent from server to vehicle units)> |
   <4: distress call> |
   <5: position report (sent from vehicle to server)>
```

Message body Free Text

```
<MESSAGE BODY free text> ::=
   <ASCII code page selection (1 byte)>
   <Length of data buffer (1 byte)>
      // the length includes this length-byte.
   {Data elements (1 byte)}
```

Message Body Standardized Message

```
<MESSAGE BODY standardized message> ::=
   <Message template number (1 byte)>
   <ASCII code page selection (1 byte)>
   <Length of data buffer (1 byte)>
   {STANDARDIZED MESSAGE PARAMETER}
```

A standardized message takes zero or more *standardized message parameter*s that represent the text that will be filled into the blanks of the message template.

Standardized Message Parameter

```
<STANDARDIZED MESSAGE PARAMETER> ::=
   <(Number of data elements + 97) && 0xFF (1 byte)>
   {Data elements (1 byte each)}
```

By adding 97, i.e. the ASCII value of lower case 'a', to the *number of data elements* we get a reasonably low *number of data elements* to be represented by a printable character, i.e. a character in the interval $['a','z']$. The maximum length of one *standardized message parameter* is still 256 bytes, including the *number of data elements* byte.

Message Body Voice Call Initiation

```
<MESSAGE BODY voice call initiation> ::=
   // this message body is empty.
```

Message Body Report Position Order

```
<MESSAGE BODY report position order> ::=
   <REPORT POSITION TYPE (1 byte)>
   <REPORT POSITION BODY>
```

Report Position Type

```
<REPORT POSITION TYPE> ::=
   <0: Send position report now> |
   <1: Send position report
       if currently within a specified area> |
   <2: Send position reports
       on specified time intervals> |
   <3: Send position reports
       on specified distance intervals> |
   <4: Send position reports
```

```
        once the vehicle is within a specified area> |
    <5: Stop sending position reports> |
    <6: Follow a specified route> |
    <7: Follow a specified route and time schedule>
    <8: Send position reports with either a specified
        time interval or a distance interval,
        whichever happens to occur first>
```

Report Position Body 0

```
<REPORT POSITION BODY 0> ::=
   // this message body is empty.
```

Report Position Body 1

```
<REPORT POSITION BODY 1> ::=
   <AREA>
```

Report Position Body 2

```
<REPORT POSITION BODY 2> ::=
   <DIFFERENTIAL TIME>
```

Report Position Body 3

```
<REPORT POSITION BODY 3> ::=
   <DIFFERENTIAL DISTANCE>
```

Report Position Body 4

```
<REPORT POSITION BODY 4> ::=
   <AREA>
```

Report Position Body 5

```
<REPORT POSITION BODY 5> ::=
   // message body is empty.
```

Report Position Body 6

```
<REPORT POSITION BODY 6> ::=
   <BOUNDING ROUTE>
```

When the vehicle has finished the specified route and arrived at its destination at the end of the route, or the vehicle deviates from the route, it will revert to position report setting number 8. Should the vehicle return to the route after a deviation, the bounding route command will be resumed.

Report Position Body 7

```
<REPORT POSITION BODY 7> ::=
   <BOUNDING ROUTE>
   <TIME SCHEDULE>
```

Once the vehicle arrives at its destination at the end of the route, or the vehicle deviates from the route it will revert to position report setting number 8. Should the vehicle return to the route after a deviation the bounding route command will be resumed. Also, if the vehicle is lagging behind or otherwise deviates from its time schedule, after a report is made to the server all subsequent check points in the time schedule will be moved by the amount of time corresponding to the deviation.

Report Position Body 8

```
<REPORT POSITION BODY 8> ::=
   <DIFFERENTIAL TIME>
   <DIFFERENTIAL DISTANCE>
```

The vehicle unit reports its position whenever the first of the specified time and the specified distance has occurred. Then both counters are reset.

Position Report Body 0

```
<POSITION REPORT BODY 0> ::=
   <POSITION (8 bytes)>
   <HEADING (1 byte)>
   <VELOCITY (1 byte)>
```

Position reports are sent only from a vehicle to the server. Similarly, the report position messages are only valid if sent from the server to the vehicles. A *position report body 0* is immediately sent in reply to a *report position body 0* request, and at a later appropriate time in reply to the other *report position body n*.

Area

```
<AREA> ::=
   <POSITION>
   <DIFFERENTIAL DISTANCE> // circle radius
   <number of corrections (4 bit)>
   {incremental distance corrections (4 bit)}
```

An *area* is used by the server to transfer the concept of a certain region, so that the server is able to ask a vehicle unit questions like "Are you in this

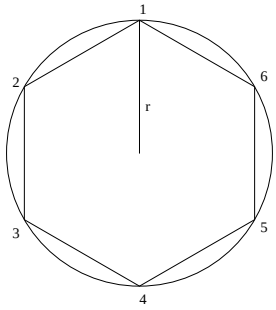

Figure 7.1: The concept of an *area* according to the communications protocol

area?" or "Let me know when you arrive in this area." by sending certain messages containing an *area*.

An *area* as it is defined in this protocol is basically a polygon approximating a circle. The *number of corrections* determine the number of points in the polygon. If the *number of corrections* is zero the polygon is in fact a circle. The incremental corrections are interpreted using a logarithmic scale.

The incremental corrections are equally distributed around the circle circumference. Each incremental correction moves one of the points of the polygon radially. The idea is illustrated in figure 7.1. The circle circumscribes the polygon. The radius of the circle is given by the *differential distance*, and the first point of the polygon being to the north, at the top of the figure. In the example illustrated in the figure, the *number of corrections* is 6, and thus the polygon has 6 corners. We will therefore also have 6 incremental corrections that each move one of the points radially. Since all the points in figure 7.1 are on the circle, we conclude that the corrections have not been applied yet, or alternatively they are all approximately zero.

Position

```
<POSITION> ::=
   <latitude (4 bytes)>
   <longitude (4 bytes)>
```

The numerical resolution achieved by representing global coordinates with 4 bytes is approximately $\frac{r \cdot \pi}{2^{8 \cdot 4}} \approx 0.005$ m in the latitudinal (north-south) direction, and $\frac{r \cdot 2\pi}{2^{8 \cdot 4}} \lessapprox 0.01$ m in the longitudinal (east-west) direction. The value of r is the mean earth radius $r = 6378137m$. A position could have been represented by 7 bytes (3.5 bytes in each direction), yielding a minimum resolution of 2.5 m, which would still be better than the DGPS sensing precision and good enough for our current purposes. This would however be too much extra programming work for a relatively small saving in the required bandwidth.

Heading

```
<HEADING> ::=
   heading256 (1 byte)
```

The current heading of the vehicle relative to the north direction. The heading measure increases clockwise. By calculating $heading256 = \frac{radianHeading}{2\pi}$. 256 we get a heading measure that fits into one byte, with a numerical resolution of $\frac{360}{256} \approx 1.4^o$.

Velocity

```
<VELOCITY> ::=
   log-velocity (1 byte)
```

The current velocity of the vehicle is encoded using a logarithmic scale. The motivation for using a logarithmic scale is that a small rounding error in the speed is less important at high speeds than they are at low speeds. The logarithmic scale will hopefully help to reduce the average relative error in comparison to using a linear scale at the same bandwidth requirement.

Assume that a speed measurement v can be transmitted using a linear scale, such as the actual speed v times a factor f. This is rounded to the nearest integer value and we assume we can use one byte to transmit the speed: $v \cdot f \in [0, 255]$. The highest speed we can transmit is thus $v_{max} = \frac{255}{f}$. The maximum relative error at a certain speed due to the limited bandwidth of the transmission is in this case given by

$$\left| \frac{(v \cdot f + 0.5) - v \cdot f}{v \cdot f} \right|$$

If instead we encode the speed measurement using a logarithmic scale such that

$$k \cdot \ln(a \cdot v + 1 \cdot b) \in [0, 255], k = \frac{255}{\ln(v_{max} + 1)} = \frac{255}{\ln(\frac{255}{f} + 1)}$$

(with $a = b = 1$ for simplicity), using the same bandwidth and the same range of measurement, we get a maximum relative error in the speed due to rounding of

$$\frac{e^{\frac{k \ln(v+1)+0.5}{k}} - 1}{e^{\frac{k \ln(v+1)}{k}} - 1} - 1 = \frac{(v+1) \cdot e^{\frac{0.5}{k}} - 1}{v} - 1 =$$

$$= \frac{(v+1) \cdot e^{\frac{0.5}{k}} - (v+1)}{v} = \frac{(v+1)(e^{\frac{0.5}{k}} - 1)}{v}$$

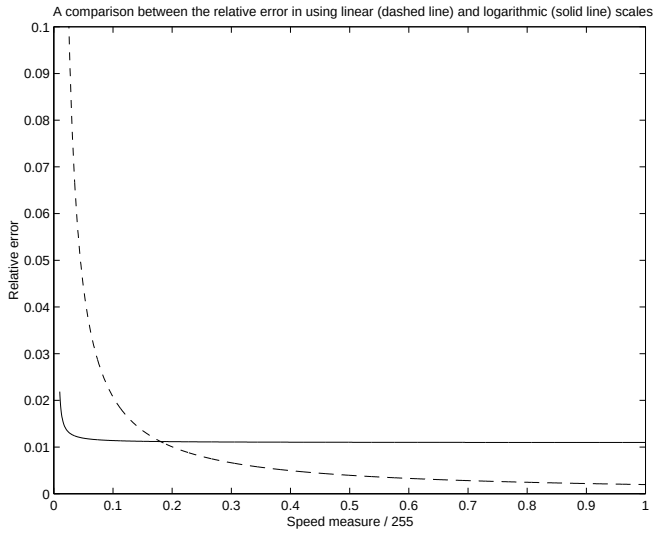

Figure 7.2: A comparison of the relative errors using linear (dashed line) and logarithmic (solid line) encoding

The relative error when using this logarithmic scale is more evenly distributed among the possible speed values than the relative error when using the linear scale. See figure 7.2 for a comparison of the maximum relative errors at different speeds. We can see that the linear scale has a very low maximum relative error at high speeds, around 1/1000, whereas at lower speeds the relative error can be large. The ideal would probably be a flat relative error curve, but the logarithmic one presented here is sufficiently flat for our current purposes.

Differential Distance

```
<DIFFERENTIAL DISTANCE> ::=
   <log-distance> (1 byte)
```

The *differential distance* is used to transfer a distance difference. It can be used to indicate a distance from a point, or as a distance interval. The interpretation of the *log-distance* in the vehicle unit is accomplished by a translation table, thus reducing number of time consuming floating point operations.

Differential Time

```
<DIFFERENTIAL TIME> ::=
   <log-time> (1 byte)
```

The *differential time* is, similar to the *differential distance*, used to transfer a time difference. It can be used to indicate a time difference from a point in time, or as a time interval. The interpretation of the *log-time* in the vehicle unit is accomplished by a translation table, thus reducing the number of time consuming floating point operations.

Bounding Route

```
<BOUNDING ROUTE> ::=
   <DIFFERENTIAL DISTANCE> // the accuracy,
                           // i.e. how wide the route is.
   <POSITION>           // where the route starts.
   <number of points>
   {<DIFFERENTIAL DISTANCE><HEADING>} // Each point is given as
                            // differential distance and angle.
```

A bounding route is a representation of a route by a sequence of points connected by straight lines and an accuracy requirement. The accuracy requirement is given by one byte that is interpreted using a differential distance translation table. By using a table, precious processing-time is saved in the vehicle unit. Then follows the *number of points* (1 byte). The start of the route is represented by a *position*, and the rest of the points are each given by a differential distance and an angle. See figure 7.3 for a graphical interpretation of this concept. A *bounding route* is calculated by the server to tell the vehicle unit how it is expected to move. As long as the vehicle unit is inside the *bounding route* is does not have to make any position reports, thus reducing the required bandwidth.

The calculation of the route is also illustrated by figure 7.3. Point A is the first point of the route. The radius r_2 is the accuracy requirement, and the radius r_1 is the accuracy requirement minus the inaccuracy of the vehicle position caused by the vehicle sensors. If GPS positioning is used, this inaccuracy will be in the order of 150 m. The apparently erratic polygon is supposed to represent the calculated route to the destination. The inner bound of the bounding route is designed to fit tightly around the route, and is extended for as long as the route stays within the bound. When the route will no longer fit, new starting points are created, B and C. After this inner bounding route has been created, the program uses the first point A as the starting point, and then calculates the differential distance and heading to the next point B, and similarly for the B to C part. The bounding route is then sent to the vehicle, and the vehicle unit will not need to report its position until it is on the outside of the outer bound.

Time Schedule

```
<TIME SCHEDULE> ::=
```

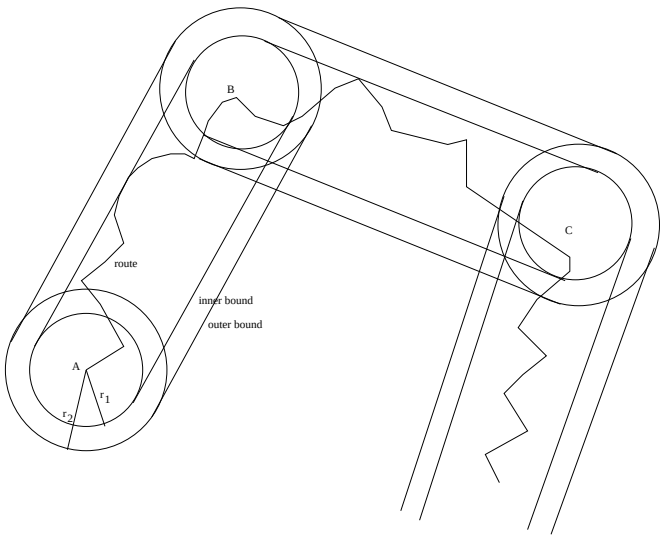

Figure 7.3: An example of a bounding route

```
<DIFFERENTIAL TIME> // the accuracy, i.e. the time tolerance
<STARTING TIME>
   // at what time the route is expected to start
// There is no need to transmit the number of points, since
// this is the same as in the preceding bounding route.
{<DIFFERENTIAL TIME>}
```

A *time schedule* is always preceded by a bounding route, and each point of the *time schedule* corresponds to the same point in the sequence of the *bounding route*. A *time schedule* is used to tell the vehicle unit when the server will expect it to arrive at the different points. If the vehicle unit realizes that it will not make it in time it will report the deviation using a *position report body 0*.

Starting Time

```
<STARTING TIME> ::=
   <time indication> (2 bytes)
```

The *starting time* is used to indicate the starting point in time of different things. The basis of the time is the time that the message was sent. The *time indication* represents the number of seconds since the message was sent. By using two bytes for the *time indication* it is possible to transfer times about $\frac{2^{16}}{60 \cdot 60} \approx 18.2$ hours into the future. This should be enough for most of our current purposes.

Message Body Distress Call

```
<MESSAGE BODY distress call> ::=
   <Distress type (1 byte)>  // E.g. airbag triggered, panic
                             // button pressed, alarm activated
   <POSITION REPORT BODY 0>
```

7.3 Message transfer procedure

This section will describe the procedure that is performed when the server has a message to send to a vehicle. First the server formats the message using the above communications protocol. Then this 8-bit per character message is converted to one or several 7-bit 160 character long messages, internally represented by strings. For each of these strings, a call is made to the outgoing message queue handler, adding the message for transmission to the telephone number of the vehicle unit. When the message handler feels that the communications device is ready for transmission, it sends the message to the device, currently a mobile telephone. The telephone sends the message in SMS format to the network operator that then internally transfers the message to a place close to the receiving mobile telephone. The local network operator transfers the message to the mobile unit. The vehicle unit receives the SMS from the server, as identified using the originating network address. The vehicle unit converts the message from 7 to 8 bit per character, and checks if it begins with "//FMM". If it does, the message is then decoded using the above communications protocol.

To sum up, this chapter has described some interesting properties of the communication channel between the fleet management server and the vehicle units. It has also defined the communications protocol and attempted to explain some selected parts of it.

Chapter 8

Positioning

This section describes how the vehicle unit keeps track of its position and how the server estimates the position of the vehicle unit.

Depending on the tracking and position-reporting mode, an estimated position of a vehicle can be calculated by the server in several different ways.

In the simplest mode, the vehicle simply appears to stay put at its last reported position until a new position is reported. When a new position report arrives at the server the vehicle representation simply jumps to the new position.

In a somewhat more advanced mode, the vehicle reports its position, heading and speed at regular intervals. The interval can be given either as a certain time difference or as a traveled distance.

Also, the vehicle unit and the server perform the same kind of calculations, so that the vehicle unit is actually estimating its own position in the same way that the server is estimating the position of the vehicle unit. Thus the vehicle unit will know when the position estimate of the server deviates too much from the real position known to the vehicle unit, and can thus send position reports in these cases. None of these modes requires a vectored map and will work on a bitmapped map, thus ignoring any knowledge of the road network.

In a simple mode using a vectored map, the average direction of the vehicle is calculated in an attempt to determine where the vehicle is headed. When the server receives a position report the vehicle is assumed to continue from that point in its average direction of travel following the most probable roads and using an appropriate ratio of the speed limit.

A more advanced mode requires the server to know the destination of the vehicle. This is reasonable in that it is often the central that tells the driver where to go. When the destination is not determined by the server operator, the driver has to indicate where he is going. If the server knows where the vehicle is going it can calculate the optimal route, and tell the vehicle unit and the driver which route it is expected to follow using a *bounding route*

message and optionally a *time schedule*. If the vehicle deviates more than a specified amount in either time or geographical position the vehicle unit will know that the server is estimating an incorrect position and can transmit the correct position of the vehicle.

The vehicle unit receives positioning information from the GPS satellites several times per second. Depending on the position-reporting mode, this information is compared to where the current-position-estimation algorithm used by the server is calculating that the vehicle is located. It is also compared to where it was on the time of the last position report, or to the limits of the bounding route. It also checks the current time to see if it is following the time schedule and has a chance of reaching the next checkpoint in time. The time is also compared to the time that the last position report was sent to the server, to see if the time interval limit has been exceeded.

This chapter has indicated what the vehicle unit does to keep track of its position, how it determines if it needs to send a position report and what the server does to estimate the position of the vehicle unit.

Chapter 9

Vehicle Unit

This chapter describes the vehicle unit and its characteristics, as well as some aspects of the vehicle unit fleet management program.

9.1 Description

The vehicle unit is basically a small box containing a computer connected to some sensors to determine the position. It is also possible to connect a GSM telephone for communication, a means for data entry from the driver, e.g. a keyboard, and a means to visualize information, e.g. a display. The display is small and limited, but it should allow the driver to read without using a magnifying glass. The two properties of the display that are of interest to the fleet management system are the number of rows and the number of columns of characters that can be displayed simultaneously. The keyboard should let the driver enter text and use the system while on the road. The processing power of the vehicle unit computer is very limited, and the program therefore avoids excessive floating-point calculations and iterative solution finding algorithms.

9.2 Vehicle Unit Responsibilities

The vehicle units in the fleet management system are basically used to communicate with the drivers, to keep track of the vehicle positions, and to inform about vehicle states, e.g. airbag status.

9.3 The Program in the Vehicle Unit

The vehicle unit program is written in the C programming language, giving the programmer full control of the computer and its limited resources, e.g. its internal memory. A drawback of using C to code the program is that C is not

object oriented, forcing the programmer to put more time into structuring the program. Another drawback is the lack of garbage collection, requiring extra care when attempting to free allocated memory. In addition, the C syntax makes it easy for inexperienced C programmers to write programs performing another task than the one intended.

As previously indicated in section 5.2, the vehicle unit program consists of

- A communication part that is responsible for sending and receiving messages, converting 7-bit messages to 8-bit messages and back, assembling messages for transmission and disassembling received messages. Message assembly and disassembly is performed using the protocol defined in chapter 7.

- A user interface. The user interface is organized in menus, some of which contain sub-menus. See section 5.2.1 for a more detailed overview of the menus. If the user makes certain selections, such as listing the incoming messages, a list-box is displayed showing one or more lines at the same time, the number of lines depending on the display properties. If the user wants to send a text message, a simple editor appears, capable of handling message templates.

- A positioning and tracking part, tracking the position of the vehicle and deciding when it is time to send position reports, based on the position report setting ordered by the server.

- A message log keeping track of all incoming and outgoing messages. The current implementation keeps all messages until the system is reset. Future versions should of course save messages until certain conditions are met. Examples of such conditions are that a certain time has passed, or that the number of messages has passed a certain limit.

This chapter has described the relevant characteristics of the current vehicle unit.

Chapter 10

Web Interface

This section describes the web interface of the fleet management server.

To help the customers of a dispatching firm in the planning of their logistics, the fleet management system is equipped with a web interface, facilitating access to vehicle data over the Internet using a simple web browser. The web interface is not intended to let the customer communicate with the driver of a vehicle. The web interface makes use of a web server that is currently under development as ISAB.

The server operator can specify which vehicles will be visible to which customers. It is assumed that this function will be used to display a vehicle to a customer, as long as that vehicle is driving towards the customer or transporting some of his goods. A vehicle is visible to a specific customer only until the final delivery on his account has been carried out, after which the vehicle disappears from her screen.

Figure 10.1 shows what the user can see in a browser. The date and time that this page was generated by the server is shown in the second line. There are three listed vehicles, but only one is being tracked by the system.

The information that is displayed about each vehicle is

ID The text on the registration plate of the vehicle, which can be used to identify vehicles in the fleet management system as well as in the real world.

Position Latitude and longitude. The two figures are in the format "dddp mm.nnnn", which is to be interpreted as:

- **ddd** degrees,
- **p** point of compass, i.e. the first character of North, East, South or West,
- **mm** minutes,

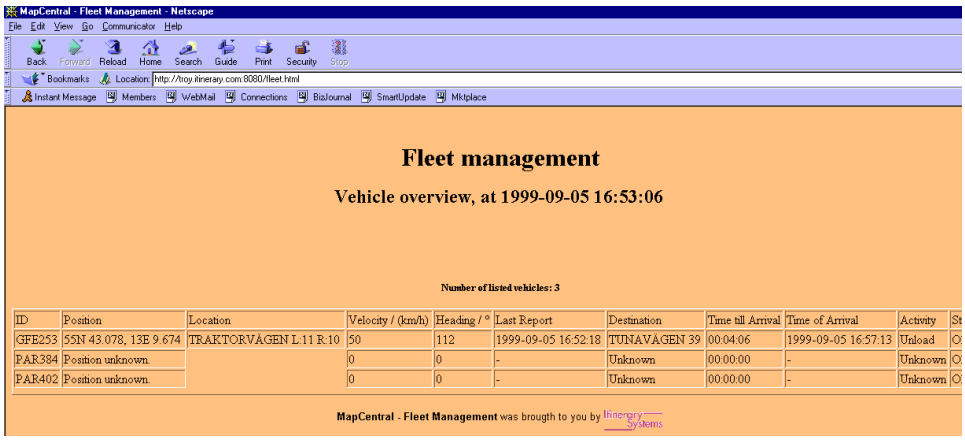

Figure 10.1: Appearance of the web interface in Netscape

nnnn milliminutes, i.e. thousands of a minute of a circle.

In the example given in figure 10.1 the tracked vehicle position would be interpreted as "55 degrees, 43 minutes, 78 milliminutes north, and 13 degrees, 9 minutes, 674 milliminutes east".

Location The name of a major nearby city, or the name of the road or street if the user and the vehicle are both in the same city.

Velocity The estimated velocity of the vehicle in kilometers per hour (km/h).

Heading The estimated heading of the vehicle in degrees. This would be zero if the vehicle were driving straight to the north. The heading increases as the vehicle turns clockwise.

Last Report The timestamp of the last position information packet that was sent from the vehicle.

Destination A string describing where the vehicle is going, or "Unknown" if the destination is unknown to the server.

Time till Arrival The time in hours, minutes and seconds until the vehicle is estimated to arrive at its destination. The current system does not have a precision high enough to make claims about the seconds, and thus one should not pay too much attention to them.

Time of Arrival The time that the vehicle is estimated to arrive at its destination.

Activity What the vehicle will do at once it arrives at its destination. This is actually the next activity, and not the current activity of the vehicle, as one might intuitively believe.

Status General status of the vehicle and its vehicle unit.

The listed information is updated on a regular basis on the server, but the web browser is instructed in the HTML code to refresh the page every 60 seconds in the current simple implementation. If the user wants more frequent updates she can press reload or refresh in her browser.

The web interface was written in C++ because the web server is written in C++. The code for generating web pages can be included in the web server at compile time. This is intended to increase the potential maximum capacity of the web-server.

This chapter has described what the web interface is used for, how the information is to be interpreted. The chapter has also given a few hints on the implementation.

Chapter 11

Simulation and Testing

This section will indicate how testing and debugging of the fleet management system has been carried out.

During the debugging and testing phases the MapCentral program is used as a server, as intended in the real system, and the vehicle unit is simulated by another standard desktop computer. The communication between the server and the vehicle unit is carried out using SMS formatted messages, but the transmission is performed over the local network instead of over the GSM mobile telephone network. This causes any performance measurements on the vehicle unit side to exceed those of the real vehicle unit. In addition, any GSM network delays have been ignored during the simulations. It is therefore not surprising that we have not seen any signs indicating that the performance requirements were not achieved. However, from the way that the vehicle unit program is written, it seems unlikely that it would not meet its rather generous performance requirements when running on the real vehicle unit computer.

Each implemented function has been tested several times after a change was made that could be expected to affect that function. As with all computer programming, there is of course still no way to guarantee a bug free implementation, but in the course of future extended actual use of the system, the concentration of bugs will decline. It goes without saying that all signs encountered so far of sluggish system behavior, crashes, lockups and memory leaks due to the fleet management system have been removed.

Due to the limited time available, the main focus has been on getting the most important parts of the system to work as intended in the simulation setup, while no actual field tests have been carried out yet. A mobile vehicle has been simulated using the MapCentral vehicle simulation. The vehicle is simulated as following the streets of a city, adjusting its speed to the speed limit, and making position reports similar to the way the real vehicle unit will make position reports. These tests have turned out in a positive way. Future field tests are expected to give indications about where additional

work should be directed.

No estimates or tests have been performed to find out if and how much, on average, the communications costs would be reduced when using this fleet management system, or how much value would be provided to the users of this fleet management system. These commercial estimates will depend, among several other things, on the size of the fleet, the skills of the fleet operators, the degree of behavioral change due to the change of the available information and the number of non-repeating assignments requiring accurate transfer of instructions.

This chapter has described how the system has been tested, and to some extent how it has not been tested.

Chapter 12

Result and Conclusion

A fleet management system program that can be used to communicate with the vehicle drivers and to keep track of the vehicles has been written. One part of the system runs as a server in a dispatching central, and another part runs as a client in the vehicle unit computer of each vehicle of the vehicle fleet. The system has been tested and verified in a simulation setup. The example application problem described in section 3.4 can be solved using the described system. The software requirement specification as defined in section 3.2 has been fulfilled, except for the automated voice connection establishment from within the server program. However, a voice communication can still be established in the old fashioned way of dialing the number of the mobile telephone manually using the keys of a telephone. No tests of the performance requirements of section 3.3 have been performed, but there have been no signs that they would not have been achieved. The design of the fleet management system makes it most suitable for small or medium sized dispatching companies. Although the server program was written to work even under Windows98, WindowsNT would be the preferred platform due to the stability problem inherent in the Windows98 operating system.

Summary

The system presented in this thesis has been designed to use a minimum of bandwidth, and thus reducing the overall communications costs. Also, the web interface will hopefully reduce the time spent communicating with customers and informing them about expected times of arrival. The web interface will also provide the customers with more frequent and more accurate estimates of times of arrivals than at least an inexperienced fleet operator is able to provide.

Suggestions for future work

What has not been done yet is a real life large-scale test, to verify that the program works as expected together with the hardware. This kind of test would also indicate whether the drivers would accept the system, and even submit ideas for future development and improvement. The automated establishment of a voice connection, as mentioned above should also be implemented, as well as an optional route guidance function.

References

Cantù, M. (1998). *Mastering Delphi 4*. SYBEX Inc., 1151 Marina Village Parkway, Alameda, CA 94501, USA.

Commercial article from G-Track (1997). Setting the pace in tracking technology. *Electron*, 14(2):21–23.

folder from Blaupunkt, I. (1998). Professionelles Flottenmanagement. Blaupunkt TravelPilot RGS 08 Professional. PPS/EDV Dispositionssoftware. Prospectus from Blaupunkt, Bosch Group, Dept K7/PMA3, Robert Bosch Str. 200, 31132 Hildesheim, Phone +49 (0) 51 21/49-40 22. Short description.

folder from Cellpoint, I. (1998). Cellpoint Systemet - en kort beskrivning. Prospectus from Cellpoint Systems AB. Short description, including some clippings.

folder from Knud Hansen, I. (1998). Euteltracs Satellitkommunikation. Prospectus from Knud Hansen Kommunikation A/S. Short description.

folder from Passo Fleet, I. (1998). So wird ein Fuhrpark dirigiert. *Verkehrs Rundschau*, 49. Distributed as off-print by Passo Fleet, as it describes their system.

folder from PTV, I. (1998). Map&Guide Fleet Monitor. Prospectus from CAS Software and PTV, Germany. Short description.

folder from SkyCom, I. (1998). WinFleet plus. Prospectus from SkyCom Telematics Systems, 32, rue des Romains, L-5433 Niederdonven, Luxembourg. Short description.

folder from TeleTec, I. (1996). TeleTrack Positioning. Prospectus from TeleTec Scandinavia AB. Short description.

folder from Volvo, I. (1999). Dynafleet Information System, GSM-version. Prospectus from Volvo Trucks Sweden AB. Short description.

H. Sannen, S. H. (1998). Real time tracking conveyances and bidirectional data communication. In *PATRAM 98, Proceedings of the 12th International Conference on the Packaging and Transportation of Radioactive Materials*, volume 3, pages 939–944.

Lippman, S. B. and Lajoie, J. (1998). *C++ Primer*. Addison Wesley Publishing Company, Corporate and Professional Publishing Group, Addison Wesley Publishing Company, One Jacob Way, Reading, Massachusetts 01867, third edition.

Mouly, M. and Pautet, M.-B. (1992). *The GSM System for Mobile Communications*. Telecom Publishing.

Papadoglou, N. K. (1998). Providing location information from GPS through GSM-short message service for the use of an AVL system. In *Proceedings of First International Symposium on Communication Systems and Digital Signal Processing*, volume 1, pages 176–180.

Stroustrup, B. (1997). *The C++ Programming Language*. Addison Wesley Publishing Company, Corporate and Professional Publishing Group, Addison Wesley Publishing Company, One Jacob Way, Reading, Massachusetts 01867, third edition.

Tegnér, E. (1810). Det eviga. In *Mindre Dikter*. Projekt Runeberg, www.lysator.liu.se/runeberg.