Mobile VTS

Project Report
Location Aware Systems (ITI45206)
”Forskning, skriving og publisering” (ITI40906)
Spring 2006
Supervisor: Gunnar Misund

Morgan Jakobsen
morgan.jakobsen@hiof.no

May 30, 2006
Horten, Norway

Østfold University College
Mobile Applications Group

Abstract

Keywords: Mobile Applications, VTS, Midlet, WMS

This paper examines the possibility of having a mobile extension to a VTS system, in the sense that a mobile device is used to receive and display live ship data, including position, course and speed. Problem areas looked into are technical limitations, usability issues and the question of who will benefit from such a product. A functional prototype is developed, and experts in the area are interviewed and presented with the prototype.

Acknowledgements

Many people have contributed in the making of this paper: The lecturing and tutoring of Gunnar Misund has helped show the opportunities and capabilities of modern mobile devices, which is greatly appreciated. The teaching of Glenn Ole Hellekjær has also been very useful when it comes to writing academic English. Furthermore, several Navtek employees have been more than helpful in both informal talks and through the evaluation of the system. And finally, my classmates have helped me more than once in different technical problems encountered throughout the project.

Thank you!

Table of Contents

Chapter 1

Introduction

VTS (Vessel Traffic Service) systems are used to monitor ship traffic in harbors, along coastlines and elsewhere. Such systems usually provide the operators with a display showing a map and the current ship traffic situation. The displays can be available on computer screens or large screen displays in control rooms or other related offices.

This paper looks into the possibility of distributing and displaying real-time ship traffic information on mobile devices. Such a solution would enable different kinds of personnel to quickly and easily get a visual overview of the current ship traffic situation. The possibility of letting the user provide complementary information through the mobile device is also explored. A simple prototype called Mobile VTS is developed and evaluated. Due to time limitations, a user centered approach to software development will not be used, even though such an approach can be vital in order to end up with a usable product [1].

The project will look at the following areas in relation to the problem statement above:

Technical issues: Technical problems encountered during the development of the system, and also problems encountered while running the system. These issues are primarily related to the mobile device.

Usability issues: Issues regarding the users experience of the system. This can be related to the point above, but also includes problems such as small screen size, slow operation, difficulties controlling the system etc.

Market issues: The product might easily become a solution without a problem, so it is vital to determine what the product can be used for. Who will use such a product? What problems will it help solve? How can it assist in a working situation?

Due to the project size and time limitations, a number of areas will not be covered in depth. These include security considerations, evaluation of what particular devices are suited for the application and why, installation and distribution of the client application, server maintenance, different business models, marketing etc. However, some of these areas are mentioned briefly throughout the paper.

The rest of this paper of organized as follows: Section 2 provides more background information on VTS systems and mobile devices. In particular, usability issues are discussed. Some of the more relevant developer tools and testing methods are also presented here. Section 3 gives some typical use cases and proposes a design for a solution to the problem. The implementation of this design is outlined in section 4. Section 5 gives test results and results from a usability evaluation of a prototype. Finally, the paper ends with a conclusion and further work recommendations in section 6.

To get the most from this paper, the reader should have some computer knowledge and be familiar with basic computer system development. In particular, UML (Unified Modeling Language) figures are used to illustrate different aspects of the proposed system. A good introduction to UML can be found in [2]. Furthermore, basic knowledge of the workings of mobile devices and cellular phones can be useful.

Chapter 2

Background

2.1 VTS Systems

VTS systems are marine traffic monitoring systems used to keep track of vessel movements and provide navigational safety for a specific area. A VTS system provides surveillance for harbors, ports, sailing routes or coastlines. Data is collected from a range of sources, including radars, CCTV, VHF radio and AIS receivers. Some data is automatically fed into the VTS system, whereas other data must be entered manually. The VTS system keeps track of the vessels present inside the relevant surveillance area [3]. The vessels, as perceived by the VTS system, are called tracks. The tracks are registered, processed and presented graphically to the VTS operators. The graphical display usually consists of a digital chart, with tracks presented as symbols on the relevant positions. A range of different attributes can be collected for each track. Vessel position, course and speed are some of the most important attributes used to ensure safety at sea. The information in a VTS system is normally accessed through computer workstations. These can be located in an operations room, or spread across several offices or buildings.

Since access to the VTS system is normally limited to workstations, people working outside the buildings have no direct access to the system. Such an access could be useful for different kind of personnel to get information about the current vessel traffic situation, and for immediate input of observed data.

2.2 Mobile devices as small computers

Mobile devices are becoming more and more powerful, and as a consequence, they are applied to an increasing number of different tasks. In [4], several scenarios are presented where the mobile device are used in different ways to help in everyday tasks. A number of different applications are readily available to consumers, se for instance [5]. The ability to run custom applications makes is tempting to replace heavy, stationary hardware with smaller mobile devices. However, the mobile devices do suffer from some usability issues, such as a small screen and cumbersome methods of typing in new data.

2.3 Usability issues of mobile devices

Some of the usability challenges of mobile devices are outlined in [6], which argues for the use of User Centered Design (UCD) approach in order to meet these challenges. Usability tips in designing for mobile devices are given in [1]. In particular, simplicity and standardization are pointed out as important aspects. How to evaluate usability in general is presented in [7]. Some experiences and special considerations needed to perform usability evaluation of mobile applications are presented in [8] and [9]. In particular, [8] provides a list of ”macro” and ”micro” factors important to mobile device usability.

A central problem emphasized in [10] is how the mobile device can support the user doing his primary job. The issue is investigated using interviews and ethnographic methods like field observation. Ideally, the mobile device should be an integral part of the work situation. This can be complicated, for instance in situations where the users hands are busy, or if the user cannot afford to look down at a graphical display.

2.4 Displaying vessel information on mobile devices

The company Navicon has developed an AIS display for mobile devices, ”AIS Mobile” [11]. The system uses a gateway server between the AIS feed and the mobile device. On the mobile device, the user is presented with a map with the vessels presented graphically. The name of each vessel is given as a label. The user can then navigate by panning and zooming, and it is possible to bookmark places of interest. The user seems to control the application using a pointing stick on the touch-sensitive screen. Target users are ”mobile employee at the harbor but also on duty at home.” The system is able to run on both the Sony Ericsson P910 and the simpler Sony Ericsson K700. This

Figure 2.1: Screens from the AIS Mobile product from Navicon.

system provides much of the same functions as the system proposed in this paper, with the main differences being:

- Data source of Mobile VTS is VTS ship data, not only AIS data. This means that a vessel tracked by radar or other sources will also appear on the mobile device. Furthermore, information specific to the VTS system, such as docking information, can also be provided.

- Mobile VTS enables the user to input information into the VTS system.

2.5 Prototype development tools

The prototype is developed using two separate software development packages. The VTS Gateway is a .NET application written in C#. Microsoft Visual Studio 2003 IDE is used for the development. The main rationale for using .NET and C# is to ease integration with existing VTS systems, in particular Navteks Navtims VTS product. Microsoft Visual Studio helps simplify the programming job, and provides a number of useful tools.

The client application is a Java midlet, developed using NetBeans 4.1 and later 5.0, with the ”Mobility pack” add-on. A number of different emulators have been used in order to test the client application. The primary test platform has been the cell phone emulator included in J2ME (Java 2 Mobile Edition) Wireless Toolkit. Testing has also been performed with emulators from Nokia SDKs, including general Nokia series 40 and series 60 phone emulators and the specific Nokia 6230i emulator. The particular platforms used for the server and client implementations are described in section 4.3 and 4.4.

2.6 System evaluation

The prototype must be evaluated according to the goals stated in section 1.

The technical issues are evaluated by the project team internally, by going through the technical problems encountered during the development phase, and summing up the most important lessons. These might include both positive and negative aspects. Also, an evaluation is done as to the technical limitations and problems encountered when running the mobile application.

To detect the relevant usability issues, external resources are used to test the prototype, and the interaction with the system is observed. The observation data along with an interview helps reveal important usability problems. An important aspect here is that the prototype is not to be developed into a complete product, it is only a proof of concept. Therefore, it is most important to discover the usability problems that can not be easily fixed in a later version of the system.

The market issues are also examined through the help of external resources. Experts in the field of VTS and coastal surveillance are interviewed in order to get an opinion as to the potential uses and users of the product.

Chapter 3

Design

3.1 Scenarios

In order to get a picture of the needs the system is intended to fill, a few scenarios have been made. The scenarios shows typical uses of the system.

3.1.1 Scenario 1 — Follow operations remotely

A port is visited by a ship that is unusually large or difficult to manoeuvre. Sailing in and out of the harbor requires careful attention by pilots, and this ship represents a particular challenge. As the visit takes place outside normal work-hours, the port master is not at work. However, by use of the proposed system running on his cell-phone, he can follow the ships movements closely as it enters or leaves the port area.

3.1.2 Scenario 2 — Get expert opinion

A particular situation occurs at sea, inside the coverage of a VTS system. As the operators at work are unsure about the situation, they call up a more experienced operator at home. The experienced operator can use his cell-phone to examine the situation himself, and type in comments to his colleges at work.

3.1.3 Scenario 3 — Get overview in foggy weather

On a particularly foggy day, a port worker wants to get an overview of outgoing and incoming ships. By use of his PDA, normally used against the ports database system, he can get an overview of the

traffic situation at sea. He can enter comments related to particular ships in order to notify others about special situations, or simply to distribute tasks among his colleges.

3.2 Purpose of the system

As shown by the different scenarios, the proposed system is an auxiliary support system, not intended to replace any existing system. The system is supposed to provide overview and detailed information about the ship traffic situation, and also to enable the user to enter in comments. The main purpose is to support the VTS or port workers in doing their jobs.

A number of requirements for a prototype can be derived from the scenarios, with some of the most important ones listed here. Some requirements are also listed that are not directly derived from the scenarios. The design of the prototype will then build on these requirements. The requirements are named to ease referencing.

3.2.1 Primary requirements

- RP1: The system shall provide a client application capable of running on a mobile device.
- RP2: The client application shall be able to present a graphical display. The graphical display shall consist of the current ship traffic information shown as symbols displayed on top of charts.
- RP3: The client application shall enable the user to navigate by panning and zooming.
- RP4: The client application shall update the graphical display immediately upon receiving new information such as new positions.
- RP5: The client application shall update the local ship information regularly from the VTS system.
- RP6: The system shall enable the user to view detailed information about each of the currently visible ships.
- RP7: The client application shall enable the user to enter a free-text comment about a ship.

3.2.2 Secondary requirements

- RS8: The client application should be capable of running on a range of different mobile devices.
- RS9: The client application should be able to predict the ships movements according to the given course and speed.
- RS10: The client application should be able to use maps from a number of different servers.
- RS11: The client application should allow the user to store ”bookmarks”, which can later be retrieved. Each ”bookmarks” shall define a particular position and zoom scale.
- RS12: The client application should send the users comments back to the VTS center and merge it into the VTS track database.
- RS13: The client should be able to display labels associated with each vessel on screen. The label shall contain at least the name of the vessel.
- RS14: The client should be able to display a line for each track on screen, representing the speed and course of the track. The line should point from the center of the track symbol and in the same direction as the track is moving. The length of the line should be proportional to the speed of the track.

3.3 Design guidelines

A few guidelines are used in the design and development of the Mobile VTS prototype. The purpose of the guidelines is to help create a good and extensible solution.

- Modular design: The prototype might need to be modified in different ways, and a modular design makes it simpler to perform these changes. The modular design is also useful if the prototype is to be extended into a real product.
- Take into account that the system should be able to run on a variety of different mobile devices, including different screen resolutions.
- Remember the user. Usability issues are perhaps more important on mobile applications than on standard PC applications [6, 10].

Also, the guidelines given in [12] are used to help create a good solution for the mobile client.

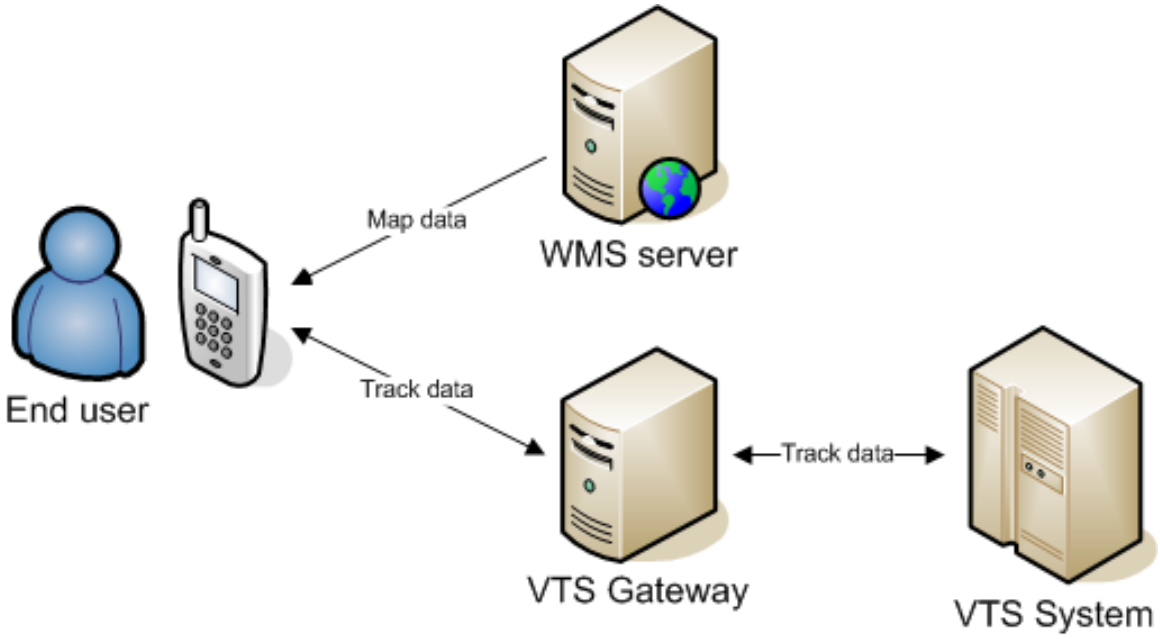

Figure 3.1: System overview.

3.4 Overall design

The main design consist of a server module and a client module. The client module is the ”client application” as defined in the requirements. The server module, called VTS Gateway, is a gateway between the VTS system and the clients. The client application receives track data from the VTS Gateway, and map data from a separate map server, as shown in figure 3.1. The track and map information is then presented graphically.

Figure 3.2 shows an overview of the system shown in UML. The map server and VTS system are existing software, and not part of this project.

The designed system provides a fast and simple VTS display, taking advantage of widespread mobile devices such as cellular phones or PDAs. It features simple installation and usage, and a very high level of availability, as the only requirement is a mobile device and antenna coverage. In fact, the system can be run from anywhere in the world.

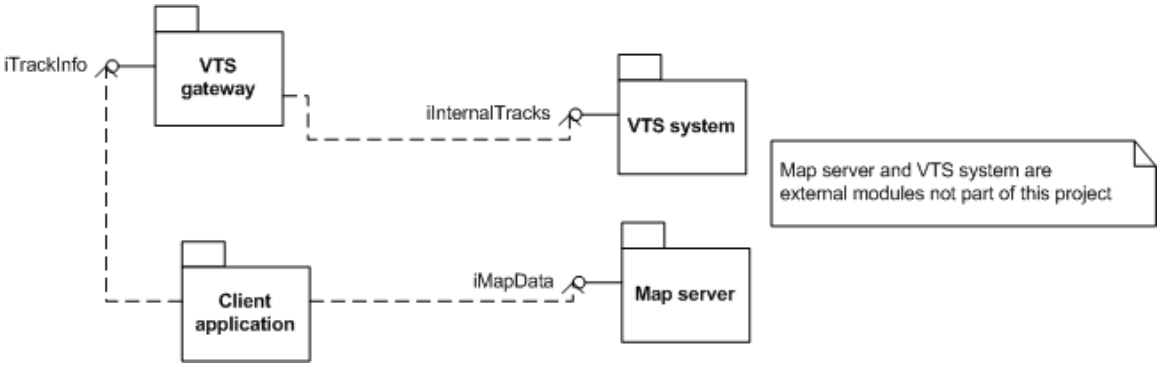

Figure 3.2: System overview with modules and interfaces.

3.5 Interfaces

The relevant interfaces in the system are described briefly below. The client application communicates with the servers through the internet, which means that the client device must be able to connect to the internet. This can be done using a number of different methods, depending on the characteristics of the client device. However, in most cases GPRS will be used. In designing the communication interfaces, it is an objective to follow well-defined standards.

When it comes to the design of the structure of the messages, the following guidelines have been used:

- The messages should be easy to create (by the sender) and parse (by the receiver). Easy means that it should be simple to implement for the programmer, not require much computer processing power, and not require much memory.

- The messages should be human readable and understandable. This helps in debugging the communication.

- The messages should be easily extendable so that they can contain more data fields.

- The communication should be stateless. The messages should be designed so that the remote party need not know anything about the transmitter in order to fully understand and act upon the message.

3.5.1 iMapData interface

Used by the client to receive map data. The client is able to communicate with OGC Web Map Services (WMS) [13]. Communication is done using the HTTP protocol [14].

3.5.2 iTrackInfo interface

Used by the client to receive track data, and to send track information typed in by the user. Communication is done using the HTTP protocol [14]. Track data is only sent upon request by the client. The request should contain the current display area of the client, enabling the server to only send the tracks that will be visible on the client screen.

The prototype client implementation requests fresh track data from the server approximately every twentieth second.

3.5.3 iInternalTracks interface

This interface is used to get and update track information from the VTS system. The interface is defined by Navtek, and communication details is implemented through the use of Navtek software libraries. Communication is done using TCP.

3.6 VTS Gateway design

The purpose of the VTS gateway is to be a gateway between the VTS system and any number of clients, as shown in figure 3.2. The main tasks are to transmit fresh track information to the clients, and to handle track information received from the clients.

The VTS gateway has two communication lines: iTrackInfo is used to communicate with the clients, and iInternalTracks is used to communicate with the VTS system.

A local database of all known tracks is maintained by the VTS gateway. Track information is regularly updated through the iInternalTracks interface. The local track database is then used as a basis for updating the clients.

The VTS gateway will run on a standard Intel-based computer running Microsoft Windows.

3.6.1 Modules for different VTS systems

A modular system is used to enable easy implementation of communication with different VTS systems. The VTS_Module component contains all functions necessary to maintain a local track

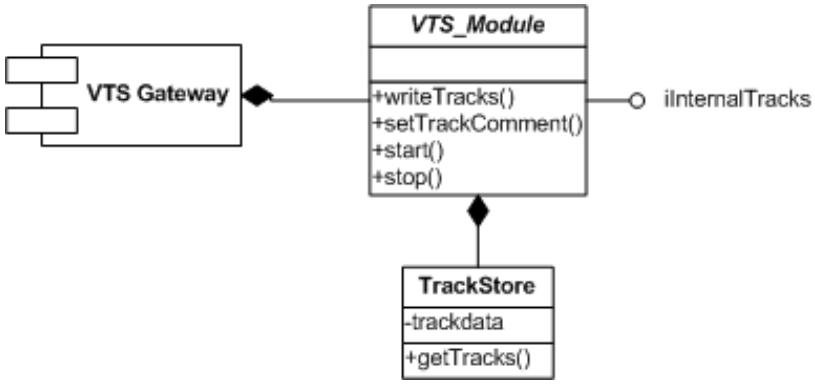

Figure 3.3: VTS_Module as part of the VTS gateway.

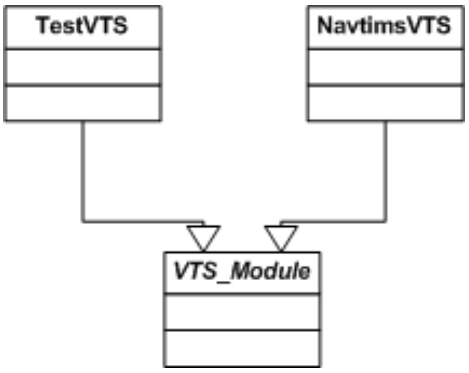

Figure 3.4: Two implementations of the VTS_Module.

database and to communicate with the VTS system (figure 3.3). This means that the VTS_Module component also contains the iInternalTracks interface.

For the prototype, two different VTS_ Module implementations are developed (figure 3.4). The NavtimsVTS implementation is able to communicate with the Navtek Navtims VTS system. This module depends heavily on Navtek software libraries. The TestVTS module is used for testing only. It does not communicate with any VTS system, but simulates a set of tracks.

3.6.2 Communication with the clients

The iTrackInfo interface used to communicate with the clients is a HTTP server allowing a large number of clients. A general principle is to keep the communication simple and avoid any kind of states. This simplifies the implementation and means that the VTS gateway does not have to keep track of each client.

All communication is initiated by the clients as HTTP requests. Two types of requests can be received: a request for updated track information, and a request containing additional information about a track.

When a request for track information is received, a list of tracks is constructed from the local track database and transmitted back to the client. The client request contains a parameter defining the geographical area visible on the client screen. This parameter is used to filter out which tracks are sent to the client. Only tracks that will be visible on the client screen are transmitted.

When a request containing additional track information is received, the additional track information is added to the relevant track in the local track database. In addition, messages are transmitted on the iInternalTracks interface in order to update the track database of the VTS system.

3.6.3 Handle multiple clients

The VTS gateway is able to handle a large number of different simultaneous clients on the iTrackInfo interface. This is done by handling the different clients in parallel, always giving room for new clients.

Furthermore, the communication protocols are designed so that the server does not have to keep track of each individual client. The protocol is completely stateless, in the sense that the server need not know which client sends which request. This concept of statelessness will probably have to be abandoned if a security scheme including authentication and authorization is implemented.

3.6.4 Communication with the VTS system

The iInternalTracks interface is used to communicate with the VTS system. The interface is a part of the VTS_Module component, and the specifications depends on the particular VTS system implemented. The interface is used to receive fresh track information from the VTS system, and to transmit additional track information entered in by the client application users.

3.7 Client application design

The main purpose of the client application is to receive map data and track information through a network connection, and to present this information graphically. In addition, the client can present more detailed track information, and allow the user to enter in supplementary information that will be transmitted back to the server.

The client application is made to run on a number of different types of mobile devices. In order to achieve this, the application is implemented as a Java Midlet. More details on this is given in section 4.4.

In order to make the application simple and portable, the user interface is designed to run on quite simple devices. This means that no pointing device is available, so the navigation mechanisms used in [11] is not possible. Furthermore, the design is made to meet the requirements of devices with ”one-handed” keypads (as described in [12]) which means that the keypad is not a full QWERTY-style. Using the findings of [10], which observed that in a working situation, the users hands were often busy, one can argue that for a mobile application intended for such use, a ”one-handed” system is preferable.

The user interface is divided into multiple screens, as recommended by [12]. One screen is displayed at a time, and each of these provide access to one or more actions. Central in the application is the map screen, presenting the map and tracks.

3.7.1 Map screen

The map screen, shown in two different modes in figure 3.5, provides the user with a vessel traffic overview picture and enables the user to access the main menu and to view track details. The screen shows a map, and ships are shown as symbols on top of the map. The handling and presentation of tracks are further described in section 3.7.2.

Map data is received upon request from a WMS map server [13] through the iMapData interface.

Data transfer status is shown as symbols on the map screen, as described in 4.4.2.

Different map modes

The map screen can be in one of two different input modes: pan mode or select mode, as shown in figure 3.5. In pan mode, the user can navigate in the display, so that information about other geographic areas are shown. The navigation consist of panning in four directions and zooming in or out. In select mode, the user can select one particular track in order to get more detailed information about it. A selector is shown around the currently selected track. The user can move the selector to the other tracks on screen, and through the press of a button he can request additional information on the selected track. The user can switch between the two modes at any time.

The rationale behind the two modes is to reduce the number of different keys needed to use the

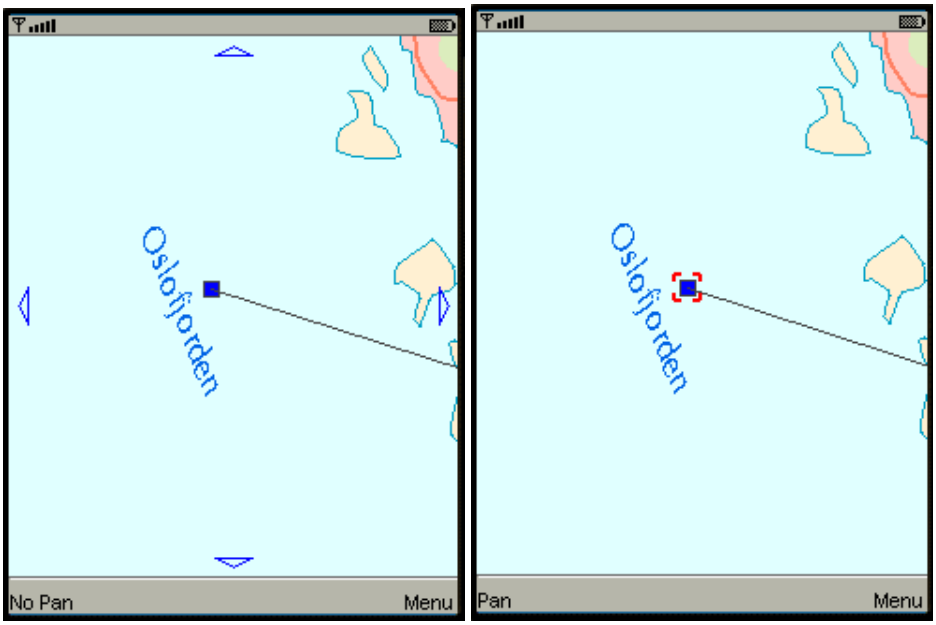

Figure 3.5: Pan mode (left) and select mode (right).

application, while at the same time allowing the user to do many different actions. In particular, directional keys are a natural choice for user input in both selecting and panning. The four directional arrows shown in pan mode (figure 3.5) are used primarily to indicate the pan mode, and to show the user that he can move in all four directions.

Navigating in the map

Navigating in the map can be split into two basic operations: zooming and panning. The user can zoom in, which means that a smaller geographical area is shown, and he can zoom out, which means that a larger geographical area is shown. Panning means to change the geographical area shown on the screen by moving in one of the four directions north, east, south or west.

In addition, the user can navigate by jumping to defined bookmarks.

3.7.2 Track handling

The client keeps its own temporary database of all known tracks. This track database is presented graphically on screen. Information in the track database is updated from the VTS gateway through the iTrackInfo interface.

Track information updates from the VTS gateway is filtered geographically, so that only tracks

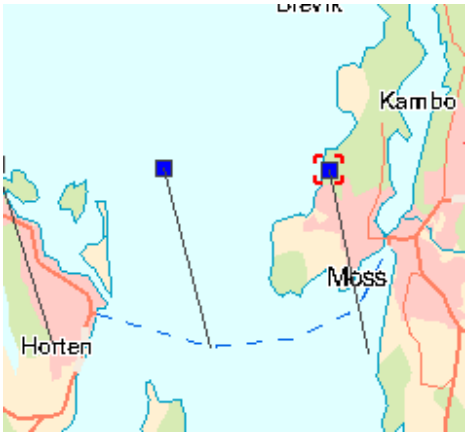

Figure 3.6: Screen with track display.

visible on the client screen is transmitted to the client. This means that the client track database usually contains only a subset of all tracks in the VTS system. To avoid potential problems related to states, lost messages and related issues, all information on all relevant tracks are transmitted on each update from the server. The client application discards all existing tracks in the local database when new information is received.

The position of each track is predicted to their assumed current position before they are presented on screen. This means that the tracks will move smoothly on screen. The prediction takes into account the latest known position of the track, the time of the latest known position, and the course and speed of the track.

A track is presented on screen as a small symbol located at its predicted position. A speed vector is drawn from each ship symbol as a line, as described in requirement RS14 in section 3.2.2. The line represents the course and the speed of the ship, where the end of the line shows the location of the ship at time N, assuming that it keeps the current course and speed. (N is set to ten minutes for the prototype implementation). Figure 3.6 shows an example of the display of tracks. The tracks are shown as blue squares, with the speed vector as a black line.

The client application is able to present further track information textually, and the user can type in additional information. This information is transmitted back to the VTS gateway through the iTrackInfo interface. The VTS gateway then integrates the information into the VTS system track database. In the prototype implementation, the only additional information that can be input is a free-text comment. Figure 3.7 shows the screen with textual track information and the text field where the user can type in his own comment.

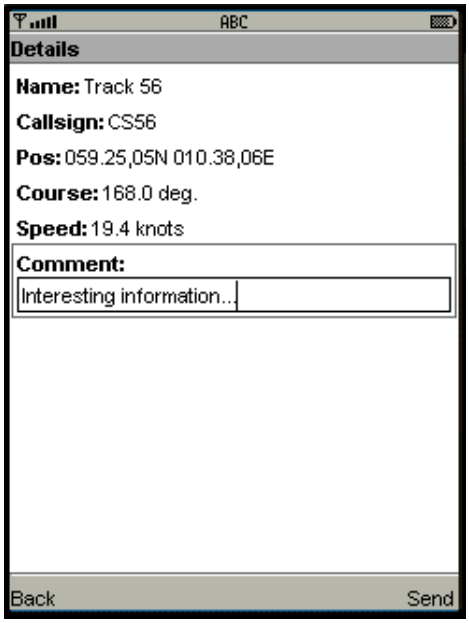

Figure 3.7: Screen with extended track information.

3.7.3 Bookmarks

The client is able to store and retrieve bookmarks. A bookmark represents a given map area as shown on the screen, where map area is defined by its center position and the zoom level. Each bookmark also has its own name. The idea behind the bookmarks is that the user can store his favorite views, and easily jump between them. The user then seldom has to do manual panning and zooming, which can be cumbersome.

Menu items and screens are provided to enable the user to write and restore the bookmarks, as shown in figure 3.8. The bookmarks are stored in a persistent storage on the mobile device.

3.7.4 Navigating between the screens

Figure 3.8 shows the different screens and how the user can move between them. Each screen is shown as a state in an UML statechart. The main menu is central in using the different functions of the application. It enables the user to perform some map functions, such as zooming, and also gives the user access to system options screen and bookmark handling screens.

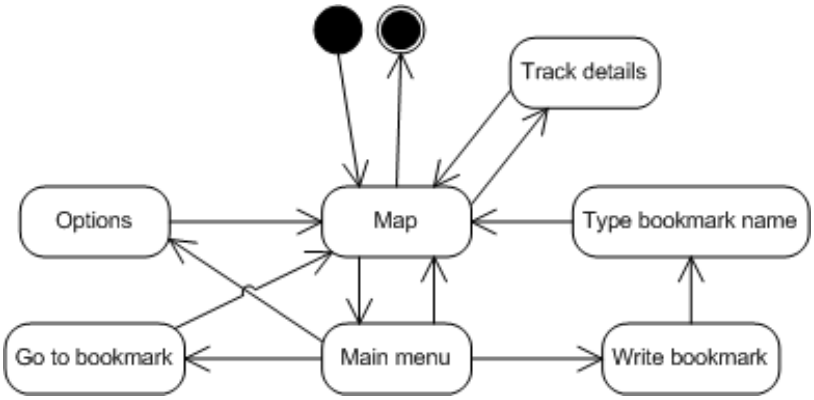

Figure 3.8: The different screens and how to navigate between them.

3.7.5 Communication

The client application has two main communication interfaces, namely iMapData and iTrackInfo, as shown in figure 3.2. The communication on both these interfaces will take place in the background, without interrupting the user. This means that the user does not have to wait for communication to complete before he can perform further actions.

Both interfaces use the HTTP protocol, which means that the client must poll for new data. On the iMapData interface, this polling will take place whenever the user performs a pan or zoom action, in which case a new map must be retrieved from the server. The client application will continue to use the current map data while waiting for new data, allowing the user to keep navigating while the map data is downloaded.

For the best user experience, the system will delay for some time before actually requesting a map from the server, as shown in figure 3.9. If the user performs more actions during this pause, the request is further delayed. This means that a map will not be downloaded until the user has stopped navigating, avoiding downloading of intermediate maps. The rule has two main positive effects: less network traffic, and the user does not have to wait for an intermediate map to finish download before the final map is downloaded. The downside is that the user, when he has finished navigating, has to wait a little extra before the map request is performed.

Updating the local track information from the server is done both as response to user action and on regular time intervals. Since the track information is filtered according to the area visible on screen, a request for fresh information must be sent after the user has done map navigation. This is done by always sending track information requests in parallel with map data requests. In addition, the client requests track information from the server on a regular time interval.

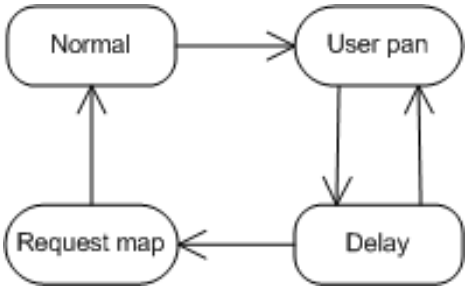

Figure 3.9: Pan process: a delay is made after panning, before requesting new map data.

When the user has entered additional information on a track, this information is immediately sent back to the server. The information will then appear in subsequent track data updates from the server.

3.7.6 Smooth operation

In order to enhance the user experience, it is important to avoid letting the user wait. For this kind of applications, the most critical parts have been identified as network communication and graphics handling. Attention must be paid to the implementation of these areas, in order to get a smooth performance for the user. An important method in coping with these problems is implementation of threads that can perform operations in the background while the user keeps working. This means that different operations are handled asynchronously in the background, perhaps invisible to the user.

Chapter 4

Implementation

This section describes some of the implementation details, with emphasis on the more challenging parts. Further details can be found by studying the source code directly.

4.1 Prototype limitations

The prototype will not contain all the functionality that should be present in a real product. However, all the primary requirements listed in section 3.2.1 are implemented. Of the secondary requirements listed in 3.2.2, requirement RS8 (Run on different phones), RS9 (predict track position), RS11 (Bookmarks), RS12 (Send comment back to VTS center) and RS14 (Speed vectors) are implemented.

Due to limitations in the mobile devices used (in particular the Nokia 6230i), it was not able to request data from the WMS map server directly. Therefore, the request is routed through the VTS Gateway. This problem is further described in section 5.6.3. The routing is not described either in the design or implementation descriptions, as it is seen as an intermediate solution for the prototype.

Note also that figure 3.8, showing the screens of the client application, does not precisely model the behavior of the prototype, it models a proposed final solution. In the prototype, the options screen is shown immediately on application startup, and is not reachable from the main menu. The reason is that during development and testing, the server IP address often changed, which meant that the settings had to be modified. This will not be the case in a completed product, where the same server should be running continually.

There are also other areas of the system that have been somewhat neglected in the development of the prototype. These areas include thread synchronization, validation of incoming data from

network and user, error handling and reporting, testing etc. However, the system works well enough for a prototype, and the time limitation and workload made it necessary to omit these areas.

4.2 Server-client communication

The communication between the client application and the VTS Gateway server takes place on the iTrackInfo interface. This section describes the communication process and the messages involved. The messages are further described in appendix A.

Communication on the iTrackInfo interface is based on the HTTP protocol, with requests and corresponding responses. Two different kinds of requests are defined:

- Track update
- New track data

The handling of these two types of requests is described below. Note that in the messages, all decimal numbers use "." as decimal separator.

4.2.1 Request for track update

This request is sent when the client wants to receive new track information. The request is sent as a HTTP GET request, with an optional bounding box parameter.

The bounding box parameter has the same characteristics as the bounding box parameter of the WMS map request, i.e.: "bbox= <west boundary>, <south boundary>, <east boundary>, <north boundary>". Each boundary value is a decimal number giving the number of degrees. If the parameter is present and valid, only tracks that are currently inside the bounding box area are returned. If no bounding box parameter is given, or the parameter is invalid, all tracks are returned.

The response is sent as plain text, with MIME type "text/plain". The data for each track is given on a separate line. Each line then consist of a number of data elements separated by semicolon. The data elements are in correct order:

- Track number.
- Time stamp for the track position. The time is given as an integer number value representing the number of milliseconds since 1. January 1970, 00:00.
- Track position, latitude. Decimal number representing number of degrees.

- Track position, longitude. Decimal number representing number of degrees.
- Track course. Decimal number representing course in degrees.
- Track speed. Decimal number representing number of meters per second.
- Track name. Free text, but can not contain ";" or linebreak.
- Track callsign.
- Track comment. Free text, but can not contain ";" or linebreak.

The message format is also described in appendix A.2.

The client should discard all previous track data when new data is received. This is done to maintain the "stateless" principle given in section 3.5.

4.2.2 Request with new track data

The client can also send a HTTP request containing new or updated information about one or more tracks. This request is a HTTP POST request, with the request data given as plain text and the MIME type set to "text/plain". The data can contain information about several tracks, with one line for each track. Each line is separated into data fields by semicolon, ";", and consist of two data fields:

- Track number
- New track comment

The message format is also described in appendix A.3. In the prototype, the only data element that can be updated is the track comment.

The response to this request is always a HTTP return code 204 - "No content". It is not considered necessary that the server responds with the resulting track data. This would only lead to complication of the communication and not add much value. If track data should be reported back, the client would have to drop the rule defined above, about discarding all existing track data when new data was received. The client should request track information when the client finds it necessary, and it should not receive track information outside this schema.

In the prototype implementation, the client will never send updates on more than one track at a time. The client also immediately adds the new comment to the track in the local track database when a new comment has been typed in by the user.

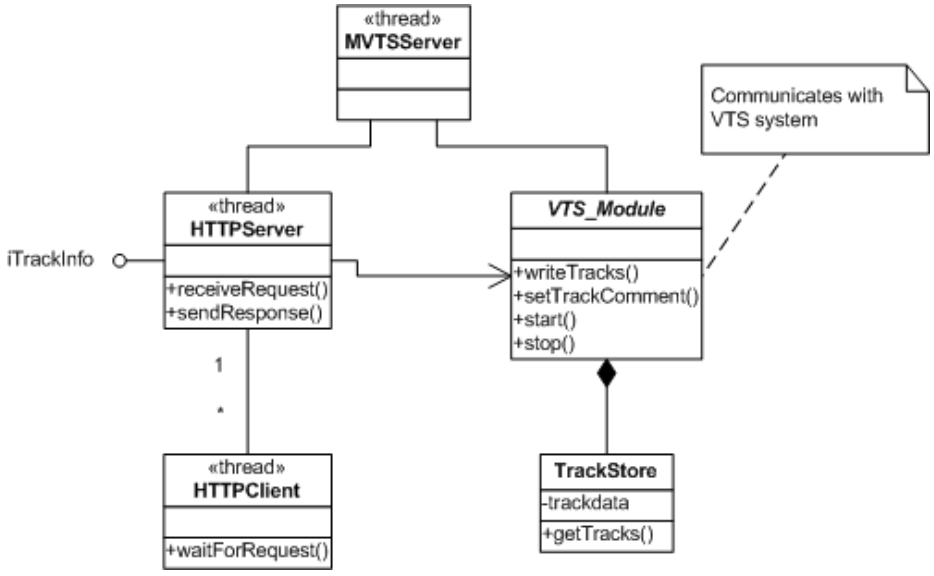

Figure 4.1: Main structure of VTS Gateway

4.3 VTS Gateway

The VTS gateway is written in C# using Microsoft Visual Studio 2003. This means that the VTS Gateway is a .NET framework 1.1 application. These choices are made primarily to ease integration of the VTS Gateway into the Navtek VTS system, and ease the use of existing libraries.

The design and main classes of the VTS gateway are shown in figure 4.1.

4.3.1 VTS system modules

As described in section 3.6.1, the VTS Gateway features a modular system in order to easily support different VTS systems. The VTS Gateway application decides at startup which one of potentially many available VTS_Module implementations will be used. A command-line parameter is used to define which module to use.

The selected module is loaded at run-time, which means that it is not linked statically to the VTS gateway application. For the prototype implementation, the VTS gateway application is not linked statically to NavtimsVTS, and it can compile and run without the Navtek software libraries used by NavtimsVTS. The TestVTS implementation is linked statically into the VTS Gateway application.

An interface is specified that each VTS_Module must implement. The interface is declared as follows:

```
public interface ITrackStore {
    void writeTracks(StringBuilder sb, GeoBox gb);
    void setTrackComment(int trackNum, string comment);
    void start();
    void stop();
}
```

Start() and Stop() are used at startup and termination. Initialization and connecting should be done in Start(), and closing down connections, freeing resources etc should be done in Stop(). writeTracks() is used to write all tracks inside the given GeoBox into the given StringBuilder, using the format specified in section 4.2.1. setTrackComment() is used to set the comment field of a track.

4.3.2 HTTP server

A simple HTTP server is implemented to handle the iTrackInfo interface. The implementation is not a full HTTP server as specified in RFC-2616 [14], but it contains the functions necessary for handling the clients in a sensible manner.

A separate thread is created for each connected client (shown as class HTTPClient in figure 4.1 and figure 4.2). In addition, there is a listener thread waiting for new clients (shown as class HTTPServer in figure 4.1 and figure 4.2), and a main thread. New object instances are also created for each new client and new request. The use of multiple threads and objects provides a separation between the different connections and modules. This means that the server will most likely run smoothly even if one of the client has connection problems or behaves strangely in other ways.

In accordance with HTTP 1.1, a connected client might send several requests on the same TCP connection. This is accounted for with the following rules:

- If a client sends ”Connection: close” in the HTTP request header, the connection is shut down immediately after sending the response.
- If the client does not sent ”Connection: close” in the HTTP request header, the connection is kept open and the client thread is ready to receive further requests on the socket.

Figure 4.2 shows the sequence of a new client connecting and sending a HTTP request to the server. Notice how a new thread is created and then handles the request. After completing the first request, the new thread waits for further requests from the same client.

A weakness in the current implementation is that the client thread does not time out while trying to receive data from the remote client. This means that unused connections and client objects are

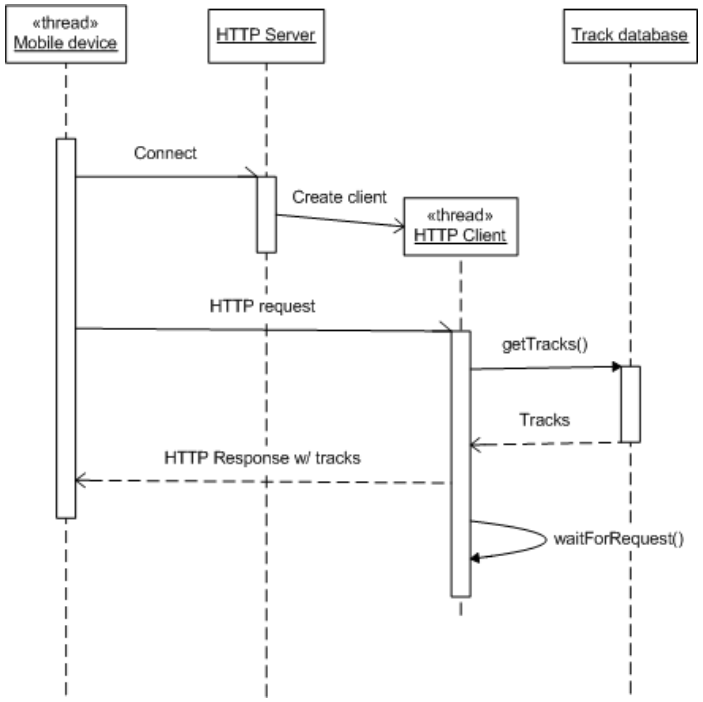

Figure 4.2: Client connects and sends HTTP request.

kept alive. Experience shows that connections are handled differently on different client platforms. Some clients always sends ”Connection: close” in the HTTP request headers. Others keep the connection alive for some time before closing it down, even if it is not used more than once.

4.3.3 OTA application provisioning

As an experiment, a simple OTA (Over The Air) application provisioning server was implemented [15]. This means that the server is able to distribute the Mobile VTS client application automatically through the iTrackInfo interface. The process involves the following steps:

- A client requests midlet information through a specific URL (`http://<servername>/midlet`).
- The server responds by sending the .jad (Java Application Description) file as MIME type ”text/vnd.sun.j2me.app-descriptor”.
- The client presents the user with the application description, and asks if the application should be installed.

- If accepted by the user, the client sends a request for the .jar file.
- The server responds by sending the .jar file as MIME type ”application/java-archive”.
- The client then installs the application.

Note that all steps taken by the client is handled automatically by the mobile device — it is not implemented by this project. The OTA application provisioning is further discussed in section 5.6.6.

4.4 Client application

The client application is implemented as a Java J2ME CLDC 1.1 [16] MIDP 2.0 [17] Midlet. CLDC 1.1 and MIDP 2.0 provides all the functions required by the client application, including support for a graphical presentation, network communication and floating point numbers. These standards are also widespread enough so that the client is able to run on a wide range of mobile devices, from cellular phones to PDAs.

The client module is an application that run on mobile devices. Care is taken to meet the special challenges regarding both implementation [12] and usability [6] on such devices.

The prototype client application can communicate with one WMS server, namely the Arealis WMS server [18]. It is easy to modify the client to use other WMS servers, and menus could also be provided to enable the user to select WMS server manually.

4.4.1 Classes and modules

Figure 4.3 gives an overview of the classes used in the client applications. The figure is an simplification, where each class in most cases represents several classes or a module.

TrackHandler is responsible for receiving and drawing tracks, as well as presenting the track details and allowing the user to type in a comment. The Track class contains data about one track. MapHandler is responsible for receiving and drawing map data. VTSScreen represents the main screen that the user looks at, with map and tracks. The BookmarkHandler class is responsible for handling of the bookmarks, including reading and writing to a permanent storage. Separate threads are used for TrackHandler and MapHandler in order to enable asynchronous downloading of data.

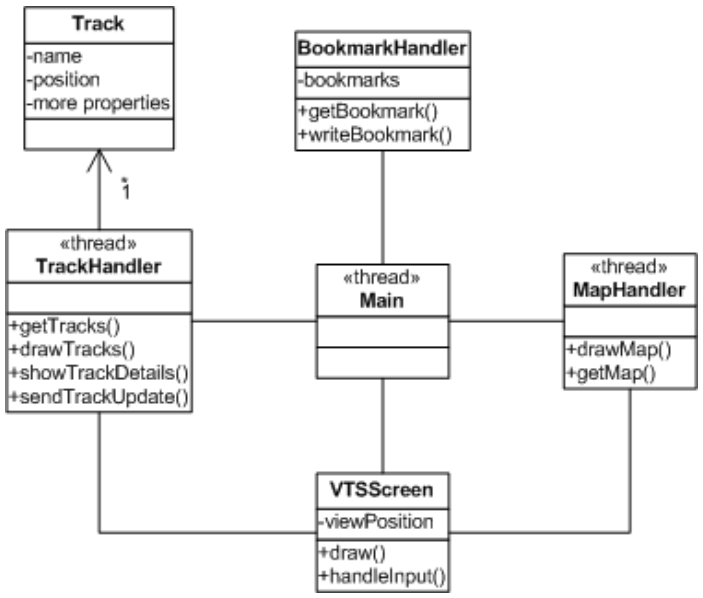

Figure 4.3: Client class structure.

4.4.2 Communication indicators

In order to inform the user of the status of the network communication, two indicators are put on the map screen, as shown in figure 4.4. The letter ”M” indicates map data transfer and the letter ”T” indicates track data transfer. When no data is being transferred, the indicators are removed. Colors are used to show the status of the data transfer: blue means that the process is ongoing, red means that some error has occurred.

This feature enables the user to quickly discover any network or communication error. The indicators have been very useful during the development process, but it is assumed that end users will also benefit from them. Data transfer problem can be caused by a wide range of issues, including incorrect setup in the application configuration, incorrect connection settings on the mobile device, lack of mobile network coverage, server problems etc.

4.4.3 Track selection

The user can pick out one particular track in order to see more detailed information. This selection process is implemented through a visible selector that always highlights the ”current” track, as shown for instance in figure 3.6. The selector is shown as four red corners surrounding the highlighted track. The use of red color means that it should be clearly visible against the normally blue

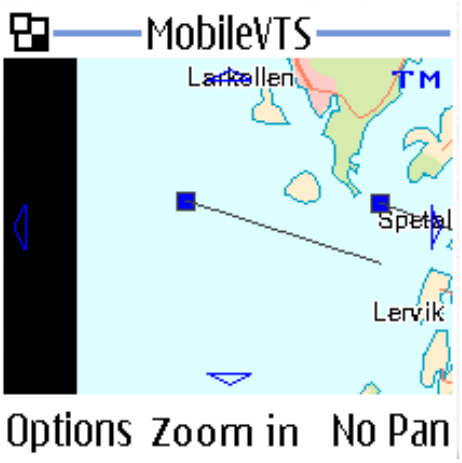

Figure 4.4: Data transfer status indicators ”T” and ”M” in the upper right corner.

or green map background. Note that track selection is not possible in pan mode.

The selector can be moved from track to track through the use of the directional keys of the mobile device. When one of the directional buttons are pressed, the following heuristic is used to decide where to move the selector: First, pick out all tracks that are at an angle less than ± 45 degrees from the direction of movement. From these tracks, select the closest one using Euclidian distance ($d = \sqrt{x * x + y * y}$).

4.4.4 Threads

As described in section 3.7.6, it is important to avoid letting the user wait for operations to complete before he can continue using the application. This is particularly true for network operations. Different background threads have been implemented that can handle this problem. They perform work in the background, while the user can continue using the application.

More work need to be done in order to analyze and handle situations that can occur when more than one thread is working on the same data. Potential problem areas include deadlocks and inconsistent data due to simultaneous modifications.

Figure 4.5 shows how the background threads for communication works: The main thread delegate the request to an interface thread, which performs the actual communication with the remote server. This pattern is used by the track interface thread and the map interface thread.

The following threads are implemented:

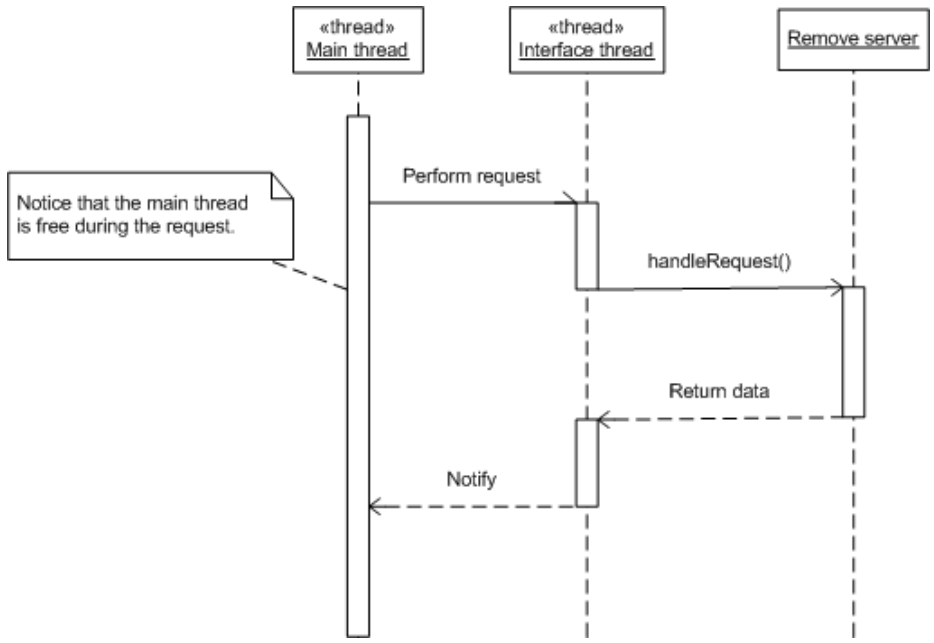

Figure 4.5: Interface thread offloading the main thread.

Main thread

This tread is the main thread that handles the user input and screen redraw. The main thread runs the HMI callbacks.

Maintenance thread

The maintenance thread is used as a timer for operations that must be performed on a time-basis. This includes the following operations:

- Screen redraw.
- Request map data — if the user has not performed any navigational operations for a predefined amount of time.
- Request new track data.

Track interface thread

This thread handles the communication with the VTS gateway. Two different operations are performed:

- Request fresh track data.
- Update track information. This is used when the user has entered in information (a comment) about a track.

Map interface thread

The map interface thread is used to request and receive map data from the map server.

4.5 Geographical calculations

Some geographical calculations are performed in both the client and server. This section describes the techniques used for the calculations.

Positions are generally stored as two real (floating point) numbers, one describing the latitude and one describing the longitude. The integer part of the numbers represents the number of degrees, and the decimal part represents the decimal degrees.

4.5.1 Screen calculations

The area that is shown on the screen on the client application is described using two parameters: the center position and the range per pixel. The range per pixel parameter is given as a real value, and contains the number of meters each pixel on the screen represents.

When a zooming operation takes place, the range per pixel parameter is modified. When a pan operation takes place, the center position is modified.

It is often necessary to calculate the pixel coordinates of a geographical position, for instance in order to position the tracks on the correct place on the screen. The calculations used are not exact, but works well for this purpose. For more precise calculations, the earth size should be used instead of the simplification that one minute is exactly one nautical mile (1852 meters) at equator.

Remember also that screen coordinates are special in the way that the y coordinate values are 0 at the top of the screen, and then increase downwards. The highest y value is found at the bottom of the screen.

We define the following parameters used in the calculations:
x, y = pixel coordinates.
lat, lon = geographical position.
clat, clon = screen center position.

rpp = range per pixels on the screen.
w and h = screen width and height in pixel.s
r = 1852*60 — meters per degree at equator (1 minute = 1 nautical mile = 1852 meters, 1 degree = 60 minutes).

Pixel coordinates can then be calculated from geographical positions with the following formulaes:

$$x = \frac{w}{2} + \frac{(lon - clon) * r * cos(clat)}{rpp}$$

$$y = \frac{h}{2} - \frac{(lat - clat) * r}{rpp}$$

4.5.2 Prediction of movement

The future position of a track is calculated through simple formulaes using the information about the current position, the course and the speed. These calculations are used in the following situations:

- On the VTS Gateway server, the position of each track is predicted ahead to the position at the time of the request. If the predicted track position is inside the filter area, the track data with the predicted position is sent to the client (see section 4.5.3 for an explanation of this behavior).

- On the client application, the position of each track is predicted ahead to the current position before they are presented on screen. This function makes the tracks move smoothly on screen, even though new information is only received every twentieth second.

- The end point of the track vector (described in section 3.7.2) is found by predicting the track position ahead ten minutes into the future.

We define the following new variables:
time = prediction time in seconds.
speed = speed in meters per second.

The following formulaes are then used to predict track positions:

$$dist = \frac{time * speed}{r}$$

$$newLat = lat + dist * cos(course)$$

$$newLon = lon + \frac{dist * sin(course)}{cos(\frac{newLat+lat}{2})}$$

4.5.3 Time problems

Initially, the intention was to transmit the track data to the client with a time stamp, so that the client could predict ahead to the current positions. However, it turned out that the time stamp was not interpreted as expected by the client. The reason is probably that the client Java VM uses a different time zone. To solve this, the track positions are predicted ahead to their current positions before they are transmitted to the client.

Chapter 5

Testing and Results

5.1 Design of evaluation test

In order to look further into the problems stated in section 1, in particular the problems related to usability and market issues, external input is necessary. This is solved through an interview and an evaluation of the prototype, performed by what can be considered as experts in the field, namely selected Navtek employees.

The test is designed using the DECIDE framework, as presented in [7]. Each of the individual steps are described below.

5.1.1 Determine goals

The goals of the test is to look into two of the problem areas listed in section 1.

1. To get input on potential usages for the product. The product should fit into a working environment, where it can assist people in doing their jobs.

2. To look at usability problems, which is problems that arise for users when they try to use the product. In this case, it is most important to look at issues that can not be easily fixed in the prototype, such as problems related to the particularities of a mobile device, including input problems, problems related to the small screen, speed problems etc. [6]. Less important usability issues are such as ordering of items in a menu, or choice of words in the user interface.

5.1.2 Explore the questions

Important questions to be answered are, for the first goal:

- What problems can the product solve?
- Who can use the product?
- How needs the product?
- How can it assist people in doing their jobs?

For the second goal, important questions are:

- What are the main problems in using the product?
- What prevents the user from doing what he wants?
- How well is the information in the system communicated to the user?

5.1.3 Choosing the evaluation paradigm and techniques

A number of different techniques are available in order to answer these questions, including observation, interview, questionnaires etc. Considering the goals of this test, elements from two different methods are used: Interview and observation. The interview part is used to get input on both goals stated above. The observation is used primarily to get input on the second goal, however it can also stimulate the participants into thinking about different usages for the product.

Usage of two different methods also help get better feedback, as the problem is approached from slightly different angles.

The interview is designed as an interview with open-ended questions. The intention is to let the participants reflect over different ideas, in particular in relation to the first goal stated above. The person running the test should try to support this kind of reflection, however, an effort must be made in order to avoid transferring his own opinions onto the participant. For both goals stated above, the questions are formulated as to avoid pointing at specific ideas or issues. For instance, the questions does not mention any usability issues directly.

5.1.4 Identify the practical issues

As the primary users of the prototype has not been identified, is seems a natural choice to test the product on expert users. In fact, to determine the primary users can be seen as a side-goal of this test! Experts in the field of VTS systems are found among the employees of Navtek.

The test should not take too much time for each participant, since their working time is valuable, and they do not gain anything from participating in this test.

The test will take place at Navteks facilities, either at a meeting room or in the participants office. A mobile device running the prototype application must be available. This device should have a large screen, as the system seems to give a better overview on larger resolution screens. Using a real mobile device has several advantages:

- It is possible to let the user test the real prototype in order to find usability problems.
- Stimulates the thoughts on how the product can be used.
- Makes it easier to come up with changes and possibilities not yet explored.
- Makes the whole test more concrete and ”real”.

The test is designed so that the mobile device is not presented immediately, but only after the first questions. This is to avoid locking the user into the same trail of thoughts that was applied during the design of the prototype. The user should be able to think freely, without being led into one specific track.

The client application must be connected to a running VTS gateway server. The server should be connected to a Navtims VTS system, with live data. Preferably, the VTS system should receive data from the Norwegian coastline AIS chain, which provides live information on hundreds of ships along the Norwegian coastline. The client would then receive and present this data during the test.

Since only one person is available to carry out the test, it must be designed with this limitation in mind: Only one person is available to run the entire test, including asking questions, instructing the user, observing the user and and collecting data.

Due to time limitation, the test will only be run on a few people.

5.1.5 Decide how to deal with the ethical issues

The participants are informed about the intention of the test, and how and what data is collected. The collected data must be presented in the report in such a way that each person remains anonymous.

5.1.6 Evaluate, interpret and present the data

Data collection is done by taking notes during the test. Use of a tape or video recorder can provide more information, but for this project it would require too much time to go through and interpret the data afterwards. Directly after testing one person, the notes must be written again so that the result is a clear and understandable text. During this process, different thoughts and interpretations can arise. These should also be written down.

The results from all tests are then evaluated, interpreted and presented in the report. The test results are qualitative data, in the form of notes. These are categorized and summarized, and the results are described below.

5.2 Carry out the test

A form was created which outlined the test process, including questions and information that was to be presented to each participant. The information included background and introduction to the project and prototype, and also information about how data would be collected and used. The form also included questions and some specific tasks that the participants was to perform on the prototype. By use of a form, one made sure that all necessary information was provided to all participant, and that all participants performed the same tasks and answered the same questions. The form can be found in appendix B.

The test was carried out as described section 5.1. The testing was done at Navteks facilities, where three different Navtek employees participated. These people represents a range of backgrounds from system development to sales.

A VTS Gateway was set up with live AIS data, as described in section 5.1.4.

Originally, the mobile application was intended to run on a Nokia 6280. However, the mobile was subject to an accident that broke the backlight of the display. Therefore, one participant ran the test on the Nokia 6280 with a broken backlight, and the other two participants used a Nokia 6230i. Notes were taken during the testing, and these were typed into a computer immediately after. This helped clean up the notes, and recall information that might not have been noted during the test.

The first question was regarding previous use of mobile applications. It turned out that all participants had used WAP, but little else. Most participants found that the primary advantage of WAP is the good availability. However, they did not use it much. A few disadvantages were also pointed out: slow operation, and the fact that each screen contain little information, which means that it is necessary to go through many pages to find the information needed.

5.3 Usability

The test contained both practical tasks for the participant to carry out, and questions about the experience of using the application. Several positive aspects were revealed, but the results also pointed out some real problem areas.

5.3.1 Positive sides

The following areas were pointed out as good in the usability context. Most of these are related to the map screen of the application:

- Navigating between the tracks work well.
- Panning is easy.
- It is easy to get started and to learn the application. This is also because there are only a few function available!
- The application responds quickly and works fast. An exception is of course the data transfer times.

5.3.2 Usability problems

A lot of the usability problems were related to the use of the menus:

- The menus and the action-button function changes when switching between pan and no-pan modes, which caused some confusion. These modes are described in section 3.7.1. The problem is caused by the fact that the ”Details” menu item is removed when entering the pan mode.
- Zooming is cumbersome, since the user must first enter the menu and then select to zoom in or out. The buttons ”1” and ”3” can be used, but all participants had to be informed of this, as there is no visual indication of this function.
- The Nokia phones always creates a menu on the left function key. On some screens, this means that there is a menu with only one item: ”OK”. The problem is described further in section 5.6.1. This issue makes it difficult for the user to see the next logical step, and partly also caused the next problem.

- After typing in a track comment, two of the users selected ”Back” instead of ”Send”. This can be caused by different things: First, the user had already pressed OK after typing in the new name, and thought this was sufficient. Second, the ”Send” item was in a separate menu, and not directly visible.
- One of the users was not used to the Nokia method of typing in text, and found himself lost in the dictionary-menus for adding a new word.
- One user pointed out that bookmarks should be more easily available, the could for instance be listed in the main menu, below the ordinary menu items.

Both participants using the Nokia 6230i had some trouble using the central 5-way button. It was difficult to activate the middle button function (the action-key), and not one of the four directional button functions. Also, the middle button function is not directly visible. Figure 5.1 shows the application running on a Nokia 6230i emulator. The 5-way button is located at the top of the keyboard area, in the middle. Notice that on the real phone, there is no physically separate middle button, even though it looks that way on the emulator.

One participant probably thought the button was four-way, and pressed the up-key to activate the ”action-button” menu item. Another participant pointed out that the problems with the 5-way button would probably go away as he got used to the device.

One test was carried out on a device with a larger screen, which caused two other problems to arise:

- The cursor became very small, making it difficult to locate.
- The map loading time was noticeable longer. The load time is probably directly proportional to the number of pixels on the screen.

One participant was slower than the others when he was panning around in the map. This meant that the intentional delay of map loading as described in section 3.7.5 caused more trouble than gain. The user had to wait out the load delay for every pan action, as he seemed to wait for the map to load before he moved on.

5.3.3 Summing up the usability

It seems that most of the usability problems has to do with the menus, whereas usage of the map screen seems to work well. Many of the menu problems are related to the layout of the menus,

Figure 5.1: Mobile VTS map screen on a Nokia 6230i

or the mapping of menu items to buttons. This is handled by the mobile phone itself, and not the application. See section 5.6.1 for a further description of this issue. Most of these problems can be solved by rephrasing or reorganizing of the menus.

The client application should run on a more advanced device than the Nokia 6230i, and on a larger screen. Problems related to the 5-way button is specific to this device, and will probably disappear on other devices with other keyboards.

5.4 Suggested new functionality

All participants came up with ideas for new functions in the system. The ideas are listed here, even though ideas for new functions were not a primary target for the evaluation.

New functions related to the map screen display:

- Show the name of the selected track. The name can be shown next to the track, or on top or bottom of the screen. Eventually, all names can be shown next to the tracks, as on a VTS workstation. The Mobile AIS presented in [11] features such labels, and they are also specified in requirement RS13, see section 3.2.2.
- In order to reduce map transfer time, maps can be simpler, or they can be cached.
- Colors can be used to show the ship size, or to indicate the age of the data.

New functions related to locating tracks:

- Put bookmarks on ships. When going to the bookmark, one would automatically jump to the particular ship. The user would then create his own list of favorite ships.
- Search for ships. The search could be performed by the server, and the client could automatically pan to the location of the ship.
- Create alarms related to track behavior. One could for instance be notified if a track crosses a user defined line.

New functions related to track information:

- Show pictures of tracks. The pictures can be retrieved from specialized internet services.
- When a note is written, it is sent through the VTS system and to the boat itself, using the AIS text message system.

- More data fields on each track. Relevant fields include MMSI number, tonnage, size etc.

5.5 Potential areas of use

A number of different potential areas of use came up during the testing. The different fields of use have been divided into three categories: port employees, secondary users and public usage. For each of the different usage areas listed here, the system would have to be modified and extended to match the specific needs of the user groups. Note that the ideas here might not be practically feasible, they are simply a summary of the feedback from the test participants.

5.5.1 Port employees

The Mobile VTS system can be used by a number of different groups of port employees, and for slightly different purposes:

It can be useful for VTS operators, especially in smaller ports. The system can be used for instance if the operator has to walk around to make security checks, or when he is off duty.

Pilots can also have use for this product. The pilots have different methods of getting information, including radio communication with the VTS operators, visual observations and use of AIS. The pilots also often have a laptop that can be plugged into all types of AIS transponders and receive information from it. This way, they have their own AIS display. However, the Mobile VTS system can be a useful addition to this.

Port workers can use Mobile VTS to get an overview of what ships are coming in. The system can also be used to divide work amongst the different workers or work teams, almost like the computer system of a taxi central.

People working in port administration and leadership can use the Mobile VTS as a method of getting an overview and check out what is happening. One test participant mentioned that a web interface showing the VTS information was very appreciated by the administrative personnel using it, even through it was not strictly required for their jobs.

5.5.2 Secondary users

Public offices and different types of companies can be seen as secondary users of live VTS ship information:

Different public authorities could use information about where ships are, when they will arrive etc. These authorities include the police, customs office, immigration office and probably more. Mobile VTS could also be used as a cooperation tool to help coordinate and share information.

VTS service personnel could use the Mobile VTS as a simple and quickly available VTS system diagnostics tool.

Shipping agents and ship-traders (people selling supplies to the ships) could use the Mobile VTS as a tool to check the location of ships and when they are expected to enter the port. These people are often traveling, and would benefit from a mobile solution.

5.5.3 Public use

The system could also be used by the general public:

Mobile VTS can be used as a simple and cheap on-board system for leisure boats. It is not a replacement for a real AIS transponder, but can be useful in order to locate the larger ships in the vicinity. For this use, the system would benefit from knowing the users own position. The position could be retrieved from a GPS receiver, or even through the use of antenna positioning, if the accuracy requirements are low.

Along the Norwegian coastline there are a lot of people who take great interest in boats, and they often follow the ship traffic that passes by. The Mobile VTS system, using the Norwegian coastline AIS chain as source data, could be a useful tool for these people, providing exact information on ship location, speed, destination, size etc. This could also be useful for people who have relatives or friends at sea, as they could easily locate and follow the relevant ships.

5.6 Technical issues

A number of practical issues have arisen during the development of the mobile client application. These issues have had different consequences: Some have led to difficulties in the development process, and some make the application perform inferior compared to the initial design goals. Also, a few positive surprises have shown up.

5.6.1 Nokia menu layout

In general, the mapping of menu items to keys are performed by the mobile device. The application does not control this directly. However, it does indicate what kind of action the menu item will trigger, such as ”OK”, ”Cancel” or ”Exit”. The mobile device then maps the menu item to the relevant

key on the mobile device, according to some heuristics used throughout the menus of the actual mobile device. This means that the application is not able to control directly how the menus and controls will be organized, and that they might look different on different devices. Figure 5.2 shows how the track details screen is handled differently on different platforms. However, by following the design guidelines, the finished product should work well on all different devices [12]. Nevertheless, some problems do occur, as described below.

Most Nokia mobile phones features two ”soft-buttons”, one to the left and one to the right, and also an action button in the middle. The particular functions of these buttons vary with the context, but the following guidelines are used: The left soft-button is used to open a menu. The action button is used to select an item, or perform some action on it. The right soft-button is used to exit or cancel an action. The same mapping approach is also used when running Java midlets such as Mobile VTS, as shown to the right in figure 5.2.

However, a few problematic situations can occur. A MIDP Form consist of several controls, such as checkboxes, textual input fields, labels etc. Nokia devices use the action button to activate or modify the selected field. One often wants to have two exit methods on a form: OK and Cancel. The Cancel function is mapped correctly to the right soft-button. The OK function is however put in a separate menu, which is accessed through the left soft-button. This means that the user is not directly presented with the OK choice, he has to enter a menu to see it.

Figure 5.2 shows an example of the problem, with a custom MIDP emulator to the left and a Nokia phone emulator to the right. On the custom phone, the text field is edited directly, whereas on the Nokia phone the user must first select ”Edit” and enter another screen in order to edit the field. Notice also that the ”Send” action is hidden inside a menu on the Nokia phone.

5.6.2 Transfer speed

The transfer speed is limited by the connection mechanism used by the mobile phone, normally GPRS. Experience shows that for mobile devices with a relatively large screen, the map transfer time is quite long, which can be frustrating to the user. A number of different mechanisms can be used to avoid the waiting time, including:

- Pre-fetch map data. Automatically load map data for the areas surrounding users current view. If the user decides to pan, the map data has already been loaded.
- Cache map data. Do not discard map data that is no longer shown on screen, but store it. If the user decides to go back, the map data is readily available.

Figure 5.2: Differences in menu layout on different devices.

- Load a simple map first, and then the full map. This means that the user will have intermediate map data available quickly. For instance, a simple map can show only land and sea, without any other details such as names.

However, in other areas faster connections might be available, such as 3G. These higher transfer rates will of course help on the problem.

5.6.3 URL length limitation

On most of the simpler mobile devices, the maximum length of an URL is quite limited. As can be seen from [19], the maximum length on a Nokia 6230i is 255 characters. This is not enough to perform a map request against the Arealis map server [18]. The problem was solved by routing the URL request through the VTS Gateway server. The map request was sent in a HTTP POST request to the VTS gateway server, which then ran the map request and returned the results to the mobile device.

This is an issue that developers must be aware of when developing applications for mobile devices. Hopefully, more advanced mobile device do not have this limitation. No further experimentation has been done on this.

5.6.4 Screen size

Experimentation has shown that the client application is easier and better to use when run on a large screen with high resolution. A large screen obviously makes it easier to see everything on the screen, and a high resolution means that more information can be presented at a time. The recommended resolution is at least 250x250 pixels.

On high resolution but small screens, items on the screen can become very small. This can be a problem on a number of mobile devices. A possible solution is to make all items in the map screen scalable, so that their size is relative to the number of pixels on the screen. This means that objects such as tracks fill more pixels on a higher resolution display.

5.6.5 Connectivity

During the development process, it turned out that using the HTTP connection mechanisms available in the MIDP programming environment is easy and useful. A well defined interface provides access to these functions, and the application developer need not worry about how the device connects to the internet and how it sends and receives data. The mobile device itself determines how and when to

connect, depending on the settings of the device. This means that if fast (3G) network connections are available, the application will automatically benefit from increased transfer speed. All in all, programming the HTTP requests is easy and provides a wide range of possible applications.

5.6.6 Install and run on different devices

The project has not looked deeply into the issues of application distribution and installation. However, some work has been done in this area, and there is a well defined standard for Midlet distribution across networks. OTA (Over The Air) application provisioning is a method for installing applications on a mobile device from a remote server. An introduction can be found in [15]. A simple OTA application server has been implemented in the VTS Gateway: by entering the URL `http://<servername>/midlet`, the application can be automatically downloaded and installed on a mobile device. (Please see the user manual of the particular device to check wether OTA application installation is supported, and how to type in the URL).

The server-side implementation was not complicated, and the result is that client installation can now be easily performed on a number of different devices. See for instance [19] for an overview of Nokia devices supporting this function.

Chapter 6

Discussion, Future Work and Conclusions

6.1 Discussion

In section 1, three different areas are pointed out as focus for this paper. The following sections discuss the results for each of these areas.

6.1.1 Usability issues

The usability issues, as revealed by the evaluation test, is summed up in section 5.3.3. Many of the problems are related to the menus, as discussed above, and particularities of the Nokia 6230i. Apart from this, the main functions of the map screen, with navigation and display of tracks, seems to work well.

6.1.2 Use of the product

A wide range of different usage areas were revealed in the evaluation test, as described in section 5.5. The different uses are spread from specific job use by port employees to a more general, public usage. These ideas span across a much wider range than presumed at the start of the project, where the focus was on the professional use by port employees, as can be seen by the different scenarios proposed in section 3.1.

As pointed out by one of the test participants, the prototype lacks some functionality in order to be a complete, commercial product. However, this additional functionality is different for each of

the different usages. Therefore, it is necessary to define the target users and intended use in order to implement the functionality required.

6.1.3 Technical issues

A number of different technical issues have arisen, some more important than others. A list of major positive and negative technical concerns are listen in section 5.6. In general, none of these technical issues puts a definitive end to the idea of a mobile extension to a VTS system.

The most annoying issue for end users is probably the transfer speeds, at least when waiting for map data transfer on high resolution screens. One can only expect the screens to get higher and higher resolutions, which would increase this problem. However, the communication network speeds will probably also increase with time, since they affect many types of applications, including video and music streaming.

The Nokia menu layout problem, which also affects the usability, can be approached in different ways. The problem can be seen as a result of the users lack of experience in using these kind of applications. As the user gets used to these menus, and perhaps also menus in other applications, they will work well. A more thorough analysis of the menu structure can also be performed, in order to increase the level of usability. Finally, it might be possible to create menus from scratch, without using the MIDP framework for menus. This way, the menus will look the same on all devices, but it will most likely require quite some work.

The URL length limitation did cause problems for the prototype implementation, however the solution found can also be used in a final product. On the other hand, a real product will most likely run under different premises. For instance, a complete solution might include a separate WMS server which then could accept much shorter URLs. It would also be possible to implement a WMS server which forwarded the requests in a way similar to the prototype implementation.

6.2 Future work

As pointed out in section 1, there are some major areas that have not been covered in this paper. Work must be done in most of these areas before the Mobile VTS can become a real product. In particular, the following issues must be looked into:

- Business model, including licensing, payment etc.
- Security considerations, related to the intended use of the product.

- Running and maintenance of the server.

No definitive solution has been found as to the particular use of the product. This is an area that could be studied more closely, for instance through field studies and end user prototype testing.

6.3 Conclusion

The questions raised in section 1 has been answered in this paper, and through these answers one can conclude that it is possible to distribute and display real-time ship traffic information on mobile devices. Such a product has a number of different potential uses, as listed in 5.5, but a conclusion has not been made as to point out one specific area of use where the product would be a definite success.

A number of different usability issues and technical aspects have been identified. However, the conclusion is that the product can be implemented technically and that it can be used by people familiar with normal computers without much problems. The different issues found provide valuable guidance if a real product is to be implemented. Furthermore, most of the information is also usable for general mobile application development, especially for applications which involves a real-time aspect and network communication.

References

[1] N. Clevenger, "Usability Within the Mobile Paradigm," *Smartphone and Pocket PC Magazine*, July 2002. [Online]. Available: http://www.pocketpcmag.com/_archives/Jul02/UsabilityMobile.asp

[2] M. Fowler and K. Scott, *UML Distilled: A Brief Guide to the Standard Object Modeling Language (2nd Edition)*. Addison Wesley Longman, Inc, 2000.

[3] Wikipedia, "Wikipedia, The Free Encyclopedia - Vessel Traffic Service," 10-May-2006. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Vessel_Traffic_Service&oldid=51951826

[4] Nokia, "Mobile Location Services," 2001. [Online]. Available: http://www.nokia.com/downloads/aboutnokia/press/pdf/mlbs.pdf

[5] Vindigo Studios, "Vindigo Studios Products," 10-May-2006. [Online]. Available: http://www.vindigostudios.com/products/

[6] L. Gorlenko and R. Merrick, "No wires attached: Usability challenges in the connected mobile world," *IBM Systems Journal*, vol. 42, no. 4, 2003.

[7] Y. Rogers, H. Sharp, and J. Preece, *Interaction Design - beyond human-computer interaction*. John Wiley and Sons, Inc, 2002.

[8] K. B. Lee and R. A. Grice, "Developing a New Usability Testing Method for Mobile Devices," *Professional Communication Conference.IPCC 2004*, p. 115, 2004.

[9] K. Kiili, "Evaluating WAP Usability: "What Usability?"," *Proceedings of the IEEE International Workshop on Wireless and Mobile Technologies in Education*, 2002.

[10] S. Kristoffersen and F. Ljungberg, "Making Place to Make IT Work: Empirical Explorations of HCI for Mobile CSCW," *Proceedings of the International ACMSIGGROUP Conference on Supporting Group Work*, p. 276, 1999.

[11] Navicon, "Navicon AIS Mobile description," 10-May-2006. [Online]. Available: http://www.navicon.dk/web/normal.php?pageid=124

[12] Sun Microsystems, Inc., "Applications for Mobile Information Devices — Helpful Hints for Application Developers and User Interface Designers using the Mobile Information Device Profile," 2000. [Online]. Available: http://java.sun.com/products/midp/midpwp.pdf

[13] OGC, "Web Map Service Implementation Specification," Nov 2001. [Online]. Available: http://www.opengis.org/docs/01-068r2.pdf

[14] Fielding et al, "RFC 2616: Hypertext Transfer Protocol — HTTP/1.1," June 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

[15] C. Enrique Ortiz, "Introduction to OTA Application Provisioning," 10-May-2006. [Online]. Available: http://developers.sun.com/techtopics/mobility/midp/articles/ota/

[16] Java Community Process, "JSR 139: Connected Limited Device Configuration 1.1," 2003. [Online]. Available: http://www.jcp.org/en/jsr/detail?id=139

[17] ——, "JSR 118: Mobile Information Device Profile 2.0," 2002. [Online]. Available: http://www.jcp.org/en/jsr/detail?id=118

[18] NGU — Norges geologiske underskelse, "Arealis map service," 10-May-2006. [Online]. Available: http://www.ngu.no/kart/arealisNGU/

[19] Nokia Corporation, "Browser Characteristics in Nokia GSM Devices," 2006. [Online]. Available: http://www.forum.nokia.com/info/sw.nokia.com/id/bb61897c-ef67-4879-9c35-1aca20745f45/Browser_Characteristics_in_Nokia_GSM_Devices_v1_9_en.pdf.html

List of Figures

Appendix A

Message specifications

A.1 Introduction

The message descriptions are given in augmented BNF format, as specified in the HTTP specification [14]. A few rules are general to the descriptions given below:

```
field_separator    = ";"
int_number         = 1*(DIGIT)
decimal_number     = 1*(DIGIT) [ "," 1*(DIGIT) ]
track_text         = (TEXT) excluding (field_separator)
```

A.2 Track update sent to client

This is the format of the track data sent to the client as response to a track update request.

```
tracks             = *(track)
track              = (track_number) (field_separator)
                     (time_stamp) (field_separator)
                     (track_pos_lat) (field_separator)
                     (track_pos_lon) (field_separator)
                     (track_course) (field_separator)
                     (track_speed) (field_separator)
                     (track_name) (field_separator)
                     (track_callsign) (field_separator)
```

```
                    (track_comment) CRLF

track_number        = int_number
time_stamp          = int_number
track_pos_lat       = decimal_number
track_pos_lon       = decimal_number
track_course        = decimal_number
track_ speed        = decimal_number
track_name          = track_text
track_callsign      = track_text
track_comment       = track_text
```

A.3 New track data from client

This is the format of the data sent from the client as the user has entered in new data (a comment).

```
track_updates       = *(track_update)
track_update        = (track_number) (field_separator)
                      (track_comment) CRLF

track_number        = int_number
track_comment       = track_text
```

Appendix B

Evaluation test form

The following form was used to perform the evaluation test. The test is written in Norwegian. The text inside quotation marks was read out word by word.

TESTING AV MOBILE VTS

Intervjuobjekt, sted, dato:

1. Introduksjon

”Som et skoleprosjekt har jeg utviklet en prototype på en mobil utvidelse til VTS systemer, Mobile VTS. Det er en applikasjon som kan kjøre på en rekke ulike mobile enheter, og som primært viser kart og track.

Som del av prosjektet er det nyttig å få innspill på hvorvidt et slikt produkt er brukbar, og også om behovet for et slikt produkt. Hvem kan ha nytte av et slikt produkt, hvilke problemer kan det løse osv. Hvordan kan produktet støtte opp under arbeidsoppgaver som skal løses?

For å finne svar på dette, ønsker jeg å stille deg noen spørsmål, og du vil også få muligheten til å trykke litt rundt i applikasjonen underveis. Resultatene vil bli brukt i prosjekt-rapporten, som anonymiserte data. Jeg vil notere underveis, både svarene du gir og observasjoner om hvordan du benytter applikasjonen.

Prototypen er å regne som et konsept, og ikke som et nesten ferdig produkt. Husk at prototypen kan endres på mange ulike måter for å tilpasses mulige bruksområder, og dette er det nyttig å få innspill om. På den annen side er det ikke fullt så viktig med innspill på spesifikke detaljer, som rekkefølge på menyer, ordvalg osv., siden dette ikke er noe produkt, men kun et konsept.”

2. Innledende spørsmål

a: Har du brukt noen mobile applikasjoner tidligere? Hvilke erfaringer har du med dette?

b: Om du har noen umiddelbare tanker rundt mulige bruksområder for Mobile VTS? (Hvem skal bruke den, til hva, hvorfor osv?)

3. Testing av prototypen

”Husk at dette er ikke noen test av deg som bruker. Målet er ikke å se om du klarer oppgavene eller ikke, men åse hvordan produktet blir forstått og brukt.” La testobjektet få bruke applikasjonen på en reel enhet. Applikasjonen skal være startet fra før.

a. Kan du bevege deg litt rundt i kartet?. Pan og zoom inn og ut. Bruk gjerne menyene også.

b. Finn en av bastøfergene og se på detaljert informasjon om denne.

c. Skriv inn en kommentar på fergen.

4. Utfyllende spørsmål

a. Hva er ditt generelle inntrykk av produktet? Positive og negative sider?

b. Hva oppleve du som de største problemene under bruk?

c. Har du noen flere tanker om hva produktet kan brukes til? Hvilke brukere kan ha nytte av et slikt produkt? Hvilke problemer kan det løses? Hva vil brukerne gjøre med denne?

5. Avslutning

”Takk for at du hadde anledning til å hjelpe meg med dette prosjektet. Rapporten skal leveres i slutten av mai. Dersom du ønsker, kan jeg godt sende deg en kopi.”