NTNU
Innovation and Creativity

Fleet Management Optimisation

Sindre Soltun

Master of Science in Communication Technology
Submission date: January 2007
Supervisor: Steinar Andresen, ITEM
Co-supervisor: Per Stein, Nordisk Mobiltelefon

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Modern ICT enables companies operating fleets of units, i.e. ships or cars, to manage these assets in new ways. Keeping continuous track of where all units are located and what their current route situation is, allows for advanced fleet management optimisation routines. Usage of such routines may gain companies in several ways, some of which include: Increased efficiency, increased control of operation and improved customer feedback.

This Master's Thesis will create a fleet management optimisation software for a given, artificial company. The company operates all snow clearing trucks in its municipality, and needs such a software to operate its fleet efficiently and customer friendly. In the process of creating this application the following tasks will be executed:

- Systemising the company's function
- Proposing a system solution
- Transport optimisation
- Programming the proposed solution

In addition to this, the thesis will discuss business models in connection with the created application, and look into possible alternative business fields where companies may take interest in the proposed software solution.

Assignment given: 23. August 2006
Supervisor: Steinar Andresen, ITEM

Preface

This Master's Thesis concludes my studies for a Master of Science degree in Communication Technology, carried out at the Norwegian University of Science and Technology (NTNU).

I would like to thank professor Steinar Andresen for his help in the initial stages of this work.

Abstract

This Master's Thesis is built around the concept of fleet management, focusing on designing and implementing a solution for such a purpose. As a target domain for this proposed system snow clearing has been chosen, and it is presented as background for the system realisation.

An important feature in a fleet management system is route optimisation. Estimations based on real-world data can be used to construct more effective routes. This optimisation process is not straightforward though, as it belongs in a domain called Vehicle Routing Problems. These problems effectively becomes unsolvable for realisticly sized datasets using traditional optimisation methods, and the reasons behind this and alternative solution approaches are presented in this text.

Enhanced fleet monitoring is another target for a fleet management system, and this requires modern localisation technology. To continuously be aware of every unit's position, an accurate tracking mechanism is necessary. Such mechanisms are also presented, focusing mainly on the Global Positioning System (GPS).

To create the actual solution, a thorough design phase was necessary. The results of this process, including a requirement specification, a design model and a test plan, are included in this report.

Based on the design phase parts of the system have been implemented, such as the graphical user interfaces and communication. The main focus of the implemetation has been on the optimisation process though, and several approaches have been tested. All implentation results, including testing results based on the test plan, can be found in this report. To offer operators a clear view of the positions of the fleet's units, a part of the system will need to work as a geographical information system. This functionality has not been implemented, but its requirements are discussed as well.

To add a market perspective to this thesis a business model for a company developing the proposed solution is presented, along with a view on how the solution may affect the business model of companies that implement it into their operations.

The last part of the report presents a discussion around the proposed solution. This discussion focuses on the qualities and shortcomings of the solution, how it compares to already existing solutions in the market, and what future work is necessary for the system to be completed.

Contents

List of Figures

List of Tables

Terminology

There are some expressions that are used to a large extent in this text. These statements are listed here, and given a short explanation.

Proposed system, proposed solution	These terms refer to the system designed for this thesis. It could alternatively have been called proof-of-concept, but since it is not complete the two mentioned terms have been chosen.
Central application	The part of the proposed system intended for the central office.
Field application	The part of the proposed system intended for the field agents.
OptiSnow	This name can be seen in some tables, but should not be given much attention. It is simply a working title for the proposed solution, and was simply used as a name for the system in the implementation phase.

Abbreviations

API	Application Programming Interface
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
CW	Clarke & Wright
E-OTD	Enhanced Observed Time Difference
ESA	European Space Agency
ESRI	Environmental System Research Institute
FI	Farthest Insertion
GAP	Generalized Assignment Problem
GDOP	Geometric Dilution of Precision
GPRS	General Packet Radio Service
GPS	Global Positioning Problem
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
IDE	Integrated Development Environment
J2EE	Java Enterprise Edition
J2ME	Java Micro Edition
J2SE	Java Standard Edition
JDBC	Java Database Connectivity
ME	Mobile Entity
MST	Minimum Spanning Tree
NTNU	Norwegian University of Science and Technology
OTDOA	Observed Time Difference of Arrival
PDA	Personal Digital Assistant
SQL	Structured Query Language
TSP	Travelling Salesman Problem
UML	Unified Modelling Language
UMTS	Universal Mobile Telephone System
VRP	Vehicle Routing Problem
WLAN	Wireless Local Area Network

Chapter 1

Introduction

1.1 Motivation

The combination of modern means of communication, location and computation has made many new technological solutions possible, and they comprise almost any business dependent on or improvable by at least one them. One field where such benefits are possible is transportation, and it may benefit from all three means. Better communication could simplify the day's work, location could offer increased, typically position based, services, and computation can be used for increased efficiency. Transportation may be divided into three categories; goods transportation, service transportation, and passenger transportation. The all have one thing in common, the more efficient the transportation is, the more money the operating company may save.

The process of monitoring and increasing efficiency of transportation problems is called fleet management. The services included in a fleet management tool vary, but some of the options are: Job optimisation, driving assistance and increased control for office operators. Understanding how such systems work, and designing a possible solution for a given domain was a major factor of motivation behind this thesis. Especially since the process of optimised vehicle routing is a difficult task.

The domain chosen as the scenario for which the proposed system is intended is snow clearing. This domain is a service transportation domain. It consists of a company operating a number of trucks which can be dispatched to locations within a given geographical area on demand. On site, the trucks perform their tasks, and then move on to the next job. This domain may not be the first one that springs to mind when one hears the words fleet

management, but it is certainly an appropriate domain for a system of this kind.

There is another factor of motivation behind this thesis as well: The market factor. While designing a solution is indeed interesting, it becomes more interesting when put up against solutions already in the market. Does it stand out at all? If so; how? What could a business model for the company implementing this solution look like? These question are all interesting, and answers to them will be sought in this thesis.

1.2 Scope

Creating a complete fleet management solution is an extensive task. Existing solutions are large and complex, and to create a solution with all thinkable functionality is, in light of the time available for this thesis, unattainable. The proposed solution of this text will offer core functionality, and the main emphasis has been on optimisation. Localisation is widely discussed as well since its necessary for i.e. data estimation in the complete system, but will not be implemented since the implementation of such a GIS-part is considered too time demanding. Based on this, what has been done is the following:

- Theoretical background. A thorough presentation of the relevant theory behind both optimisation and location.

- Design. A detailed design, supported by UML diagrams, will be presented for most of the proposed solution, for the GIS parts only a thorough discussion for what the solution needs to do is provided.

- Implementation. Proposed user interfaces, the main database and the optimisation process are implemented, GISes and some other functionality are not.

- Business model. A business model for a company selling the proposed solution is developed.

- Discussion. A discussion around the proposed solution's quality compared to the scenario and other fleet management solutions completes the thesis.

1.3 Methodology

Procedure

This thesis has been carried out as a combination between a litterature study (theory and business model creation) and a software design process (design and implementation). The first weeks where used to gain knowledge about the subject of fleet management. This was used as guidance through the design and implementation process which followed thereafter. The latter stages have been used on creating the business model and (mostly) on writing this report.

References

Most references used in this text is based on scientific sources such as books, scientific articles and lecture notes. These sources should provide thrustworthy information, and as so-called "good" sources they have been used without much scepticism towards their contents. Some references point to information which should be used more carefully. This includes "help"-pages for i.e. a software, product catalogues, and other information on vendor specific sites. Such information has been used some places in the text, after careful considerations. In some cases, noteably GIS, much of the information comes from a support site for this type of program, and it links to pages on a vendor page. However some information has been used since it is also available in a text book. Other places, vendor specific references has been used in connection with presentation of used software, and in these places such links are considered acceptable (for providing depth). This does not mean however, that the author call on readers to accept everything found on these pages as legit.

1.4 Report Organisation

The rest of the chapters in this report contain the following:

- Chapter 2. Presents the scenario for which the proposed solution is intended. It will focus on a detailed, high-level requirement presentation, as well as identifying some challenges and presenting enforced scenario limitations.

- Chapter 3. Presents all relevant optimisation theory, namely the Travelling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP).

- Chapter 4. Presents relevant localisation theory. The main focus is on the Global Positioning System (GPS), but a little information on some alternatives is also given.
- Chapter 5. Presents the design for the proposed system. Identifies a requirement specification, and design the system and creates a test plan based on this information.
- Chapter 6. Presents detailed information about the implementation. The focus is both on what has been done and what needs to be done. Test results for the current implementation of the proposed system are also included.
- Chapter 7. Presents a business model for a company manufacturing and selling the proposed solution.
- Chapter 8. Provides a detailed discussion around the proposed solution. Includes discussion of quality, possible complications, similar solution comparisons, etc.
- Chapter 9. Provides a conclusion for this Master's Thesis.

In addition to this, the following appendices have been added for some additional report depth:

- Appendix A. Use case diagram for the proposed solution.
- Appendix B. Detailed UML diagrams for some of the packages designed.

Chapter 2

The Scenario

This chapter will introduce the specific scenario to which the optimisation software will be created. The focus will be on giving a general description of the scenario, discussing possible sub-scenarios and giving a thorough account of the abilities and limitations of the proposed system solution.

2.1 General Description

The scenario is centred on a company which offers snow clearing services to The City of Trondheim. The company takes care of all snow clearing in the municipality, using a number of snow clearing trucks. In its current situation, the company's way of communicating with its drivers is solely based on a general voice calling system, without any central system keeping track of where all fleet units are located. The company wishes to enhance its management by implementing an advanced communication system, allowing for more detailed control of the company's operation. Such a system should also offer the possibility to optimise the routes of each truck. The company lists the following requirements for the system solution:

- Communication between central office and field units, both text and speech, available.
- Location of all trucks and drivers continuously available on-screen in the central office, using i.e. a map service.
- Optimising the routes of all trucks both at start-up and as new jobs are added possible.

Based on these requirements, a discussion of how the system should function is presented in the next section.

2.2 Solution Discussion

In this section the different parties involved in the system, and what requirements they bring to the proposed solution will be discussed. The system will consist of two distinct parts, namely the central application and the field application.

2.2.1 Trucks and drivers

The system requirements state clearly that there should be a clear distinction between the trucks themselves, and their drivers. This is logical given that each driver may operate different trucks on different shifts, and knowing who drives each truck offers several benefits:

- Increased transparency and employee control.
- Possibility of monitoring driver performance.

While modelling and implementing driver monitoring is beyond the scope of the design presented in this report, it might well be a required part in a real-world product, especially for HR reasons. (Knowing which driver does what is likely to be a logical feature in any system of this kind.)

Apart from the obvious managerial issues in knowing what each driver does, it really is the truck which is the main asset in the field. Trucks may have different abilities, i.e. some trucks may have the ability to sand roads and areas, while others do not. The optimisation process will therefore be based on the trucks' abilities and not the drivers', and thus the field application should be located permanently in each truck (driver independent). To identify which driver operates each truck, the field agent system will need driver login functionality in connection with its start-up, creating a connection between driver and vehicle.

2.2.2 Localisation

An important part of the processing in the central application is based on the localisation of trucks and jobs. The localisation of the trucks will be fed to the central application from all field applications continuously, and this means that the field application will need to use a means of localisation to keep continuous track of its exact whereabouts. The solution for this system will be using the General Positioning System, GPS, for this purpose. Other

possibilities exist in theory, such as using GSM or UMTS triangular positioning, but maximum deviation is much higher using these methods compared to GPS. (More on location methods in chapter 4.)

For the central application no such device is needed. It does however require a typical feature in combination with GPS; a map service. The requirements state that the central application should offer its operator an on-screen representation of where all trucks are located, and the obvious way to realise this is by using a map service. By receiving the coordinates from all field applications continuously, and feeding them to the map service, an updated map should be available at all times.

For the field application its local map service may be used to find the estimated shortest path to the next job location.

2.2.3 Communication

Another central part of the system is communication. To effectively interchange information between the central office and the trucks, such as position and job status data being sent from the trucks to the office and updated job lists being transferred in the opposite direction, one needs a good communication system. The proposed system is expected to work independently of the network technolofy underneath, thus this section just presents the most likely

In the truck this will be based on a wireless communication method. The obvious choice in today's network situation is using mobile devices using either GPRS or UMTS connections to send the data. For the purpose of this system, the offered data rates in these solutions should be adequate, since information sent will mainly be limited to short information messages. In addition to this standard way of connection the device which the field application operates on could also have WLAN connection possibilities. Deployment of WLAN access for mobile devices in public areas is rapidly increasing, an example being the project "Trådløse Trondheim" ([1]) in Trondheim, Norway. Using WLAN instead of GPRS/UMTS should offer higher data rates and lower prices for subscribers, lowering the snow clearing company's communication costs.

For the central application using a standard broadband connection is the obvious choice of communication.

2.3 Scenario limitation

The core task of the system's operation will be the optimisation process. While better location and communication both could help a company run mor efficiently, the real savings most likely lie in better route management. Unfortunately route optimisation belongs in a group of problems called Vehicle Routing Problems (VRPs), and these problems are hard to solve efficiently. (The VRP is thoroughly presented in chapter 3.) Lots of factors may affect the optimisation process, and it is thus important to make an exact limitation to how the problem will be modelled. For the scenario presented here, and thus the proposed solution of this thesis, the following assumptions are made.

- Homogenous fleet. All trucks operated by the company will be of the same kind, meaning that they have the same technical capabilities and theoretically perform the same tasks in the same time. (The latter is of course also driver dependent.)
- Direction independent. It takes the same time to travel from node A to node B, as it takes from node B to node A.
- No special abilities. The trucks clear snow, and that is it. No sanding or other special abilities leading to further capacity restrictions exist.
- Only one depot. All trucks have the same location as their starting (and ending) point.
- Capacity determined by time usage. For each job there is a time-to-complete estimate, and each truck will be able to perform jobs for a given amount of time each day. This means that the only capacity constraint in the scenario is the time available.
- Distances between nodes are given as time estimates. The system will estimate the time needed for moving between all nodes, and it is these time estimates which are used as distances (arc costs) in the calculations.
- Optimisation takes place daily. The process is executed every day (typically in the morning), and thus this is no long term planning system.

This may seem like to many limitations, and in a commercial product it is likely that some of these assumptions must be shelved, and that the system surroundings will be different. For academic purposes, presenting the consept of a fleet management tool, these limitations are acceptable though.

The problem described is the VRP in its original form, with arc costs, node demand and vehicle capacities, and will make up the basis in any commercial system as well. The implication of several possible additional requirements are discussed in section 8.2.

To decide the time capacity of the trucks, some assumptions need to be made as well. Since the complete solution will not be implemented, such capacities are not hard coded into the current implementation. A possible scheme could be to set it as a fixed percentage of the total time available per truck per day. In Norway a standard day's work is eight hours, including a half an hour lunch break. If we assume that the time used for doing jobs is significantly higher than the transportation times[1], say 4:1, then this means that six hours, or 360 minutes, a day can be assumed available for executing jobs. This is of course a highly debatable figure, and for a commercial solution such values could be obtained in a much more realistic manner by simply contacting actual snow clearing firms.

The bottom line for this section is that the solution presented in chapters 5 and 6, and discussed in chapter 8, must be viewed within the terms stated here, as these limitations greatly affects its basis of operation.

[1]This is a sound assumption for a city municipality such as Trondheim, bu could be way off for a municipality in rural areas.

Chapter 3

Optimisation Theory

This chapter will present relevant theory in connection with the optimisation part of the proposed solution. The problem of scheduling trucks the most effective way presents a problem called the Vehicle Routing Problem (VRP). In addition to being used to actual vehicle routing (such as in the scenario described in this thesis), it also applies to i.e. such problems as effective newspaper printing, paper cutting, and network deployment. The VRP is known to be NP-hard , meaning that to solve this problem within reasonable time for realistic problems will require constraint relaxation or heuristic approaches, which will be thoroughly presented in this chapter. To understand some of the solution methods of the VRP, one also needs to understand the nature of the underlying problem in a VRP; finding the shortest possible tour in a bidirectional graph such that all nodes are visited exactly once. This problem is called the Travelling Salesman Problem (TSP), and will be described in the first section of this chapter.

3.1 TSP

The Travelling Salesman Problem is a widely researched optimisation problem. The job of effectively traversing a graph so that each node is visited exactly once relates to many real world situations. The first one coming to mind of course being the problem indicated in the problem name; a salesman travelling between a number of cities carrying goods from a depot at a fixed location. A mathemathical description of the problem is given in subsection 3.1.1. This problem is also NP-hard, which will be discussed in subsection 3.1.2. Alternative solution methods, mainly heuristics, will be discussed in subsection 3.1.3, to round off this section.

3.1.1 Problem description

This section will define the TSP mathematically, and is based on [2]. The goal of the TSP can be mathematically stated as follows:

$$Min \ z = \sum_{i=0}^{N} \sum_{j=0}^{N} C_{ij} * x_{ij} \tag{3.1}$$

Where N equals the number of nodes in the tour, including the depot (typically node 0), C_{ij} equals the cost (given as length, time, etc.) from node i to node j, and x_{ij} is a binary variable which takes the value 1 if the edge from i to j is included in the tour, 0 otherwise. In addition to this several constraints apply to the problem. The equations below ensure that each node in the network is visited exactly once:

$$\sum_{i=0}^{N} x_{ij} = 1, \qquad \forall j \tag{3.2}$$

$$\sum_{j=0}^{N} x_{ij} = 1, \qquad \forall i \tag{3.3}$$

These equations make sure that each node is both the origin and the target of exactly one path.

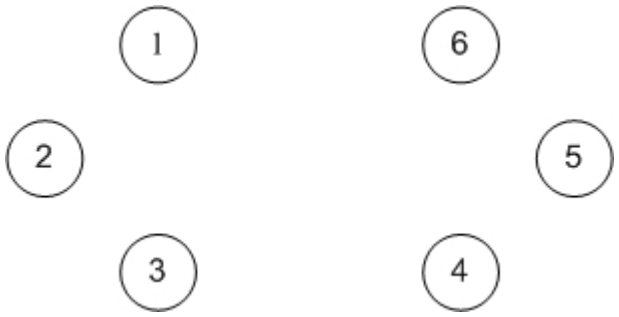

Figure 3.1: Six nodes in two distinct groupings.

There is however one more pitfall that needs to be taken care of. Consider the case where we have six nodes. Three and three nodes are close together, while the distance in between these groupings is large relative to the internal distances in each grouping. The example is illustrated in figure 3.1[1]. Now

[1]In the example used in the figures in this section the numbering starts with 1, and not 0 such as in the equations. This does not make a difference other than that the summation starts at 1 and not 0 when solving the optimisation. (The same figures will be used in modified versions in section 3.2, where a depot will be introduced as node 0.)

consider we execute the optimisation process based on this figure, and assume that the distances between the nodes in the figure describes the real life distances accurately. The result will be two tours, one consisting of the nodes 1, 2 and 3, the other consisting of the nodes 4, 5 and 6, shown in figure 3.2.

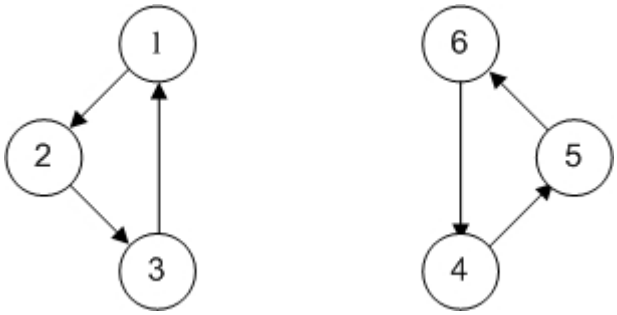

Figure 3.2: Optimal solution *with* subtours.

To take care of this subtour problem additional constraints are needed, and one way to do this is by adding the following constraint to the problem:

$$\sum_{i \in S} \sum_{j \in \overline{S}} x_{ij} \leq 1, \qquad S \subset N \tag{3.4}$$

This equation states that for each subset S in N, there exists at least one arc from a node in S to a node not in S. As long as this constraint holds we are ensured that the solution obtained when executing the optimisation will contain no subtours, and for the example given in this section, the obtained result should be as shown in figure 3.3.

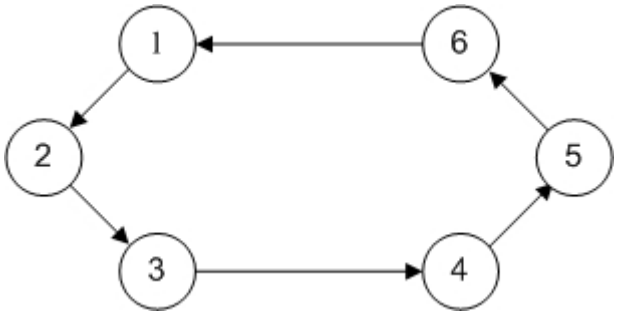

Figure 3.3: Optimal solution *without* subtours.

By combining equations 3.1, 3.2, 3.3 and 3.4, and other information given in this section, we get a complete problem statement as follows:

$$
\begin{aligned}
Min\ \ z = & \sum_{i=0}^{N}\sum_{j=0}^{N} C_{ij} * x_{ij} \\
s.t. \qquad & \sum_{i=0}^{N} x_{ij} = 1, && \forall j \\
& \sum_{j=0}^{N} x_{ij} = 1, && \forall i \\
& \sum_{i\in S}\sum_{j\in \overline{S}} x_{ij} \geq 1, && S \subset N \\
& x_{ij} \in \{0,1\}, && \forall i,j
\end{aligned}
$$

It is worth noticing that this solutions holds for both symmetric and asymmetric TSPs, but in the symmetric case the problem can be somewhat simplified because $C_{ij} = C_{ji}$.

3.1.2 Problem discussion

As mentioned in the introduction to this section, TSP is an NP-hard problem. To solve the problem can be done easily proportional to a time of (N-1)!, since the number of tours for a problem with size N is (N-1)!/2 ([2]). A more effective solution was proposed by Held & Karp in the 1960s, and a short description of this approach, based on [3], follows.

The Held-Karp approach uses a variant of Minimum Spanning Trees (MSTs), called 1-trees, to obtain the solution. A 1-tree is a graph with vertices 1..N, a tree on vertices 2..N and two edges connecting vertex 1 to the tree. To find a minimum 1-tree for a given vertex V, one calculate an MST for the other vertices, and add the two least costly edges from V to vertices in the MST. The following properties can be noted:

- A tour is a 1-tree where each vertex has degree 2. (Each vertex has two connected edges.)
- A minimum 1-tree is easily computed.
- Transforming internode distances $C_{ij} \rightarrow C_{ij} + \pi_i + \pi_j$ will not alter the TSP, but will change the 1-tree.

By systematically varying the values of π_i, it is possible, but not guarantied, that we may find a 1-tree which is also a tour. There may however be a 'gap' between the problems, and by creating a function $f(\pi)$ which is non-negative, measures this gap and is assumed to be 0 when it is possible to obtain a 1-tree which is a tour, the problem is transformed to the task of minimizing $f(\pi)$. This problem corresponds to a linear program, similar to that of the TSP but without the integer constraints. The program can be solved in several ways; Held & Karp discusses column generation, a greedy ascent method, and a branch & bound technique. For more case relevant information on column generation see [4], and for branch & bound see [5]. The solution obtained by using Held-Karp, provides a lower bound for the TSP which is very accurate, as $Z(HK) = 0.99Z(TSP)$ according to [2]. (Z is the optimal solution using Held-Karp version and the original TSP respectively.) Held-Karp is also more effective than the solving the problem the standard way, as it has a runtime of $n^2 * 2^n$, which is significantly lower than (n-1)! for large values of n.

Even though Held-Karp is more effective than the standard solver it still runs in exponential time, and thus is not effective enough for larger problems. Even worse, this 40 year old solution is still to be substansially toppled by other solutions as of today, and no breakthrough is expected. For this reason a lot of effort is put into alternative solution methods, which are not necessarily based on finding the optimal solution, but a solution wich is acceptable relative to the optimum. Such solution approaches exist in a plethora, and they exist in the form of concrete heuristics, and so-called metaheuristics, which are algorithms using several other methods, heuristics included, to find a solution. Several of the more widely deployed heuristics are presented in the following section.

3.1.3 Heuristics

Farthest Insertion

Insertion algorithms are a regularly used approach to solve TSPs. One of these algorithms, Farthest Insertion (FI), is presented here. It has been chosen as the example because it performs best of the basic insertion algorthms compared to optimum ([2]). Other insertion algorithms include Cheapest Insertion and Nearest Insertion. Insertion heuristics are based on including one new node in the tour per iteration. FI differs from other insertion algorithms in that it is not based on including the nearest node next. Instead it works as follows ([2]):

Step 0: Start with a partial tour T consisting of one node.

Step 1: Find the node k not in T which nearest node in T is farthest away. $C_{lk} = max\{min\{C_{ij}|i \in T\}|j \in N \setminus T\}$, where l is the node closest to node k.

Step 2: Replace edge (i,l) with edges (i,k) and (k,l) so that $\Delta C(T,k)$ is minimised[2].

Step 3: If all nodes N is included in T; end. Otherwise; return to step 1.

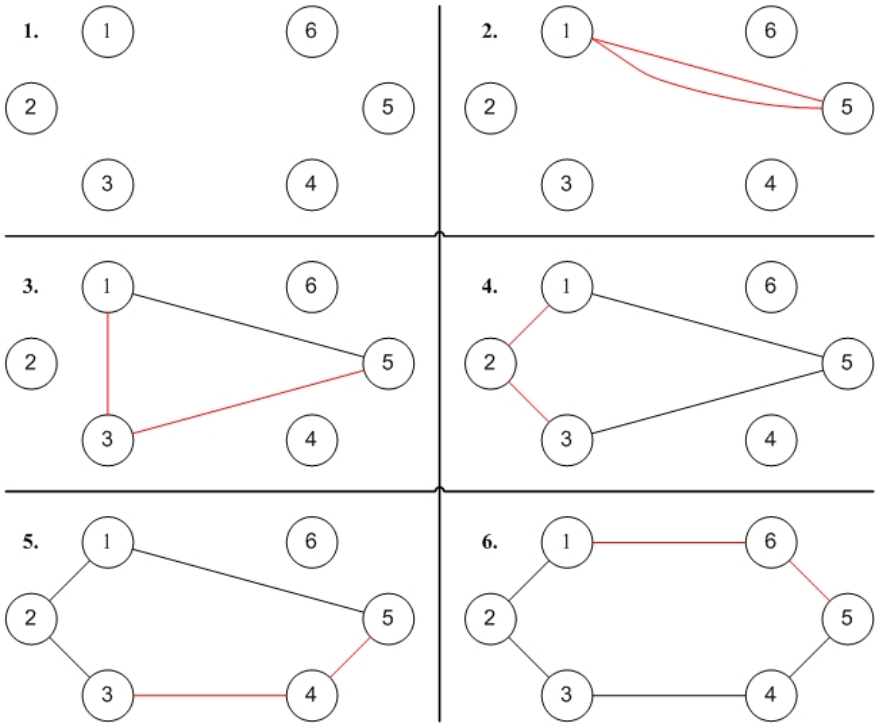

Figure 3.4: Farthest Insertion solution of sample TSP.

FI provides an upper bound on the optimal solution of the TSP, on average $Z(FI) = 1.16Z(TSP)$. While this is a pretty loose upper bound, it is solvable in polynominal time ($\Theta(n^2)$), and can be calculated in seconds for large datasets. For this reason FI is interesting with regards to the system solution presented in chapters 5 and 6. For an example on how FI will work on the example presented in figure 3.1, see figure 3.4. (The runtime for FI and other

[2]In step 2 there are two alternative ways of insertion, before or after node l, the insertion is made where the cost of adding k is lowest.

runtimes mentioned later in this section are taken from [6], which contains demonstrations and runtimes for several TSP approaches.)

Christofides algorithm

Christofides algorithm is an approximation method to solve the TSP, and it consists of the following process [2]:

Step 0: Find an MST.

Step 1: Construct a matching for nodes with an odd number of connected edges.

Step 2: Construct an Euler tour MST and matching edges.

Step 3: Construct a TSP tour by using short cuts.

To find the MST two algorithms can be used; Kruskal and Prim. These two algorithms both construct a TSP for any graph without non-negative cycles, which is usually the case for TSPs. For more on these algorithms, see [5]. The matching is done because it is only possible to construct an Eulerian tour in a graph if every vertex has an even number of connected edges. Figure 3.5 shows a stepwise use of Christofides[3]. Christofides algorithm offers a slightly

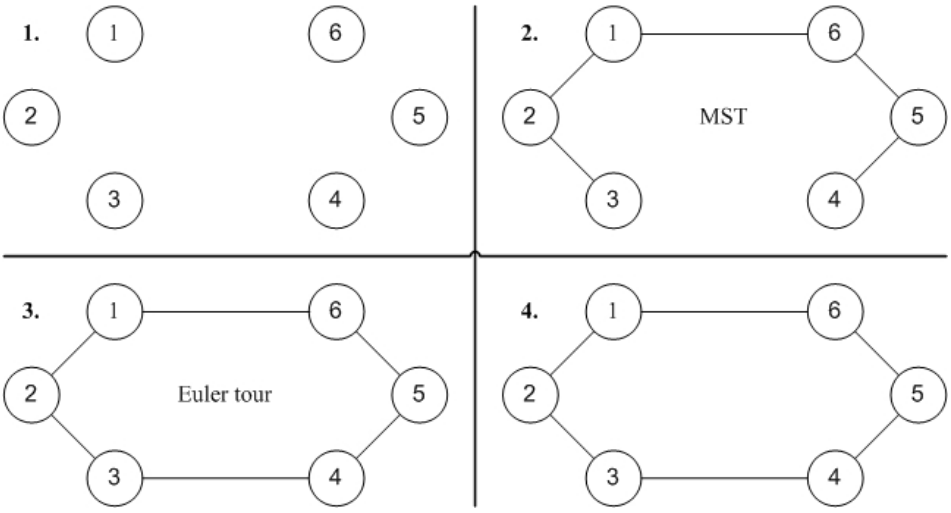

Figure 3.5: Christofides algorithm on sample problem.

better bound than FI, namely $Z(CHR) = 1.14Z(TSP)$, according to [2].

[3]While the Euler tour and the TSP tour are equal in this example, this is not generally the case.

The runtime however, is, yet still polynomial, slighthly worse ($\Theta(n^3)$) than for insertion heuristics.

Post optimisation

To enhance the heuristics even further, we can use a local search algorithm. These algorithms are based on mapping each solution f of an optimisation problem to a neighbourhood $N(f)$. $N(f)$ consists of all tours g which can be created by swapping k edges in f with k edges not in f, where $k \in \{2..n\}$. For the TSP, k usually is given the value 2 or 3. In figure 3.6 it can be seen how the algorithm works with $k = 2$, with the original tour 1-2-3-6-5-4-1 and the following cost matrix:

$$\begin{bmatrix} 0 & 3 & 5 & 10 & 11 & 8 \\ 3 & 0 & 3 & 11 & 12 & 11 \\ 5 & 3 & 0 & 8 & 11 & 10 \\ 10 & 11 & 8 & 0 & 3 & 5 \\ 11 & 12 & 11 & 3 & 0 & 3 \\ 8 & 11 & 10 & 5 & 3 & 0 \end{bmatrix}$$

Using the local search method described, called 2-opt, renders the following possible swaps and associated ΔC[4]:

$(1,2),(3,6) \rightarrow (1,3),(2,6) \Rightarrow \Delta C = 5 + 11 - 3 - 10 = 3$

$(1,2),(4,5) \rightarrow (1,5),(2,4) \Rightarrow \Delta C = 11 + 11 - 3 - 3 = 16$

$(1,2),(5,6) \rightarrow (1,6),(2,5) \Rightarrow \Delta C = 8 + 12 - 3 - 3 = 14$

$(1,4),(2,3) \rightarrow (1,3),(2,4) \Rightarrow \Delta C = 5 + 11 - 10 - 3 = 3$

$(1,4),(3,6) \rightarrow (1,6),(3,4) \Rightarrow \Delta C = 8 + 8 - 10 - 10 = -4$

$(1,4),(5,6) \rightarrow (1,5),(4,6) \Rightarrow \Delta C = 11 + 5 - 10 - 3 = 3$

$(2,3),(4,5) \rightarrow (2,5),(3,4) \Rightarrow \Delta C = 12 + 8 - 3 - 3 = 14$

$(2,3),(5,6) \rightarrow (2,6),(3,5) \Rightarrow \Delta C = 11 + 11 - 3 - 3 = 16$

$(3,6),(4,5) \rightarrow (3,5),(4,6) \Rightarrow \Delta C = 11 + 5 - 10 - 3 = 3$

[4]In the example ΔC is estimated as the sum of costs for new edges minus sum of costs for existing edges, implying that improvments results in negative ΔC values. In some literature, like [5] this is calculated the opposite way.

As can be seen from the list above, only one swap will improve the tour ($\Delta C < 0$), so edges (1,4) and (3,6) are swapped with edges (1,6) and (3,4) and the new tour is 1-2-3-4-5-6-1. In the next iteration no swaps improves the tour, so 2-opt terminates and the last tour is kept. (In this case the tour found by 2-opt is also the optimal tour.) Using 2-opt we get a tighter upper bound to the TSP, with $Z(2-opt) = 1.06Z(TSP)$. (3-opt is even better, with $Z(\text{3-opt}) = 1.04Z(TSP)$ ([2])). The runtime of 2-opt is $\Theta(n^2)$ per iteration, and should thus provide valuable improvement of the initial solution in acceptable time if the number of iteration is limited.

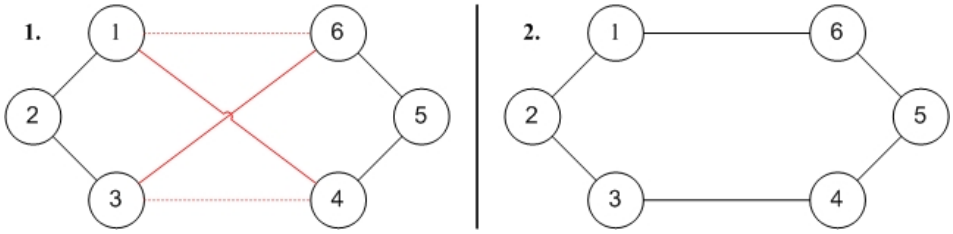

Figure 3.6: 2-opt used on sample problem with initial solution (left).

Other heuristics and approaches

Several other approaches to solving the TSP effectively and accurately exist, but will not be thoroughly presented here. The most notable of these include tabu search (see [2]), the Lin-Kernighan local search algorithm (see [7]), branch and bound techniques, and genetic algorithms.

3.2 VRP

In the VRP the problem of the TSP is extended to multiple travellers. As the TSP, which is a subproblem in the VRP, it is widely resarched as it is an NP-hard optimisation problem with widespread practical interest. The VRP adds several problems on top of those presented in the TSP section, and these will be discussed in subsection 3.2.3. As with the the TSP, heuristics are widely used to solve the VRP, and such approaches are presented in subsection 3.2.4. First, however, the basic problem formulation and description will be provided in subsection 3.2.1, followed by an alternative approach in 3.2.2.

3.2.1 Basic problem description

The problem definition for the VRP is similar to that of the TSP, but a few additions are needed. First of all the solution space needs to be extended, since the variables now need to reflect not only which nodes the edges connect but also which traveler traverses them. This leads to the following problem model ([8]):

$$
\begin{aligned}
Min\ z = & \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{j=1}^{K} C_{ij} * x_{ijk} \\
s.t. \quad & \sum_{k=1}^{K} y_{ik} = K, && i = 0 \\
& \sum_{k=1}^{K} y_{ik} = 1, && i = 1 \ldots N \\
& \sum_{i=0}^{N} x_{ijk} = y_{jk}, && \forall j, k \\
& \sum_{j=0}^{N} x_{ijk} = y_{ik}, && \forall i, k \\
& \sum_{i \in S}\sum_{j \in \overline{S}} x_{ijk} \geq 1, && S \subset N, \forall k \\
& x_{ijk} \in \{0, 1\}, && \forall i, j, k \\
& y_{ik} \in \{0, 1\}, && \forall i, k
\end{aligned}
$$

This formulation makes sure that the depot is the starting point of K edges, that all other nodes are the starting points of 1 edge, that the relationship between x_{ijk} and y_{ik} is correct (x is defined as in the TSP only per traveller, while y_{ik} is 1 if node i is visited by traveller k and 0 otherwise.), and that for each traveller no subtours exist in its path. To illustrate the problem, the same example is used as in section 3.1 and its subsections, but now an additional node (0) is introduced as the depot. The example can be seen in figure 3.7. To illustrate the problem, the same example is used as in section 3.1 and its subsections, but now an additional node (0) is introduced as the depot. The example can be seen in figure 3.7, and a possible solution to this problem if there are two travellers is shown in figure 3.8. It is common to add another constraint to the VRP as well; a capacity constraint. In the case that each traveller has a capacity of some good, and at each node there is a

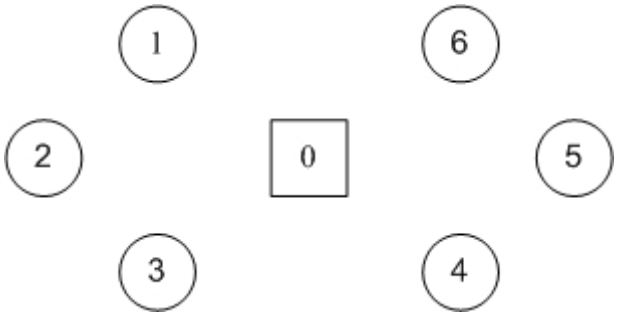

Figure 3.7: A sample VRP.

given demand for this good, the following constraint may be added:

$$\sum_{i=0}^{N} d_i y_{ik} \leq b_k, \qquad \forall k$$

If the set of travellers is homogenous then b_k could simply be replaced by a common capacity b. (This restriction, including the homogenous b_k values is a part of the traditional VRP description.)

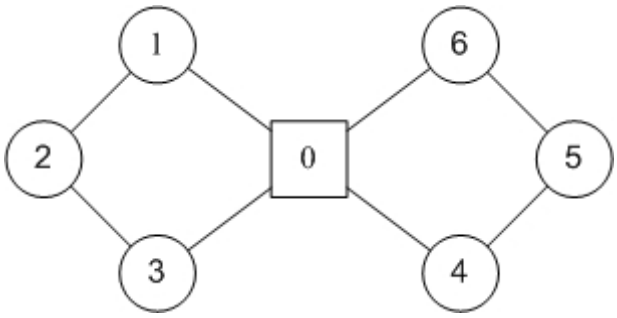

Figure 3.8: VRP solution with K = 2.

3.2.2 Alternative problem description

What can be drawn from the model formulation in the previous subsection, is the extensive amount of computing necessary even for relatively small data sets. Even more so than the TSP, the VRP has a steep rise in variables and constraints as the number of nodes and travellers increase. For a problem with 50 nodes and 10 travellers (which is a rather small problem) there are 25500 variables (25000 x-variables and 500 y-variables), 1060 non-subtour constraints, an exponential number of subtour constraints, plus the binary

constraints ([8]). Clearly this model rapidly becomes unsolvable. (Even solving this example requires a powerful processing tool.)

Another model formulation can be used to solve the VRP, which is slightly different. It is a so-called set partitioning model, and can be formulated as follows (for more info see [4] and [9]):

$$
\begin{aligned}
Min\ z &= \sum_{j=1}^{J} C_j x_j \\
s.t \qquad & \sum_{j=1}^{J} x_j = K \\
& \sum_{j=10}^{J} a_{ij} x_j = 1, \qquad \forall i \\
& x_j \in \{0,1\}, \qquad \forall j
\end{aligned}
$$

In this formulation x_j takes the value 1 if route j is used, 0 otherwise, and a_{ij} is 1 if node i is used in route j, 0 otherwise. (Note: a_{ij} is a matrix of constants!) Let us use this model to solve our sample problem for the case with K = 2 and the following distances from node 0 to all other: $\begin{bmatrix} 4 & 6 & 4 & 4 & 6 & 4 \end{bmatrix}$. If we decide that each traveller will cover exactly three nodes and no nodes where $C_{ij} > 10$ may be in the same tour, we get the following formulation:

$$
Min\ z = \begin{bmatrix} 14 & 21 & 23 & 23 & 21 & 14 & 14 & 21 & 23 & 23 & 21 & 14 \end{bmatrix}^T x
$$

$$
s.t. \quad \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix} x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} Traveller\ 1 \\ Traveller\ 2 \\ Node\ 1 \\ Node\ 2 \\ Node\ 3 \\ Node\ 4 \\ Node\ 5 \\ Node\ 6 \end{matrix}
$$

Using this matrix representation all constraints, both traveller constraints (line 1 and 2) and node constraints (lines 3 thorugh 8), are covered. The costs are calculated using the cost matrix presented in section 3.1.3, and the depot distance costs mentioned above. Solving this problem will result in two possible solutions: $x_1 = x_{12} = 1$, all other x's 0, or $x_6 = x_7 = 1$, all other

x's 0, and $z = 28$.

While this method could certainly be solved more quickly than the original problem formulation, there are several drawbacks to this solution method too. To make this method effective, it is necessary to make good reductions in the route space. This can be done in several ways, limiting the number of allowable nodes in a tour, and/or the maximum direct distance between nodes in a route (both of which is used in the example) are some examples. However this is done though, good solutions are likely to be dropped. Using the full route space turns this into a problem only solvable in exponential time, which renders it unsolvable for realistic problem sizes as well. There is another problem as well: For each route the distance needs to be found, which means a TSP needs to be solved for each route to generate usable route costs. While some of these routes may be calculated accurately as they may be relatively small, heuristics are likely to be needed here as well, adding a furher margin of error to the problem. In the example in this section the route costs have been calculated by visiting the nodes in increasing order, and all routes with length 23 in the cost array could actually be traversed more effectively (cost = 21).

3.2.3 VRP discussion

As was mentioned in the previous subsection, the VRP, even in its simplest state, is a hard problem to solve. But there are several additional constraints that may be applied to the VRP which complicate matters further. Normal additional demands to the VRP model includes ([8]):

- Tour length. The standard definition of the VRP minimises the total cost for all tours, this will lead to some tours being longer than the others, and restrictions may apply both to the allowed length of any tour and the maximum allowable distance between the longest and the shortest tour.

- Time windows. Each node may require to be visited within a specified time window, which limits its possible position in any tour.

- Node priorities. In an environment where tasks have variable importance, where there are internal reliances between the nodes, or where it is natural to use a FIFO approach, support for node priority may be necessary. This would be especially important in environments where visiting all nodes is unattainable. (I.e. demand > supply.)

- There may be multiple depots available.

- There may not be a fixed number of vehicles, meaning that the objective may change to fullfilling all tasks with as few vehicles as possible.

- Multiple objectives. The goal might be to find a solution which is satisfactory in several ways, i.e both in terms of time and resources spent.

- Assymetric costs. The costs of moving from node i to node j may be different from the costs related to moving in the opposite direction.

Explaining how all of these items affect the time demands for solving the VRP is beyond the scope of this report, but the point that should be noted is that all these items add constraints to the problem definition, which to a variable extent increases the effort needed to solve the problem. Some of these items will be further discussed in chapter 8, in relation to the discussion of the proposed solution.

3.2.4 Heuristics

The TSP approach

It is possible to solve a VRP as a TSP if a few adaptions are made. The VRP can generally be described as: "Construct the most effective tours where k vehicles starting from a depot d visits n nodes exactly one time combined." An alternative description, making a special case TSP, can easilty be made: "Construct the most effective tour where a vehicle visits a depot d exactly k times, and all the other n nodes exactly once." To model this, the cost matrix needs to be extended with one row and column per additional vehicle. These rows must have the following properties: The same cost to all other nodes as the depot, and an infinite cost between the depots (both the real one and all the virtual ones[5]. For the sample matrix with costs as given in 3.1.3 and depot costs given as in 3.2.2, the cost matrix using this approach will look

[5]In this formulation copies of the original depot are made for the sake of calculation, these will be referred to as "virtual depots".

as follows:

$$\begin{bmatrix} 0 & \infty & 4 & 6 & 4 & 4 & 6 & 4 \\ \infty & 0 & 4 & 6 & 4 & 4 & 6 & 4 \\ 4 & 4 & 0 & 3 & 5 & 10 & 11 & 8 \\ 6 & 6 & 3 & 0 & 3 & 11 & 12 & 11 \\ 4 & 4 & 5 & 3 & 0 & 8 & 11 & 10 \\ 4 & 4 & 10 & 11 & 8 & 0 & 3 & 5 \\ 6 & 6 & 11 & 12 & 11 & 3 & 0 & 3 \\ 4 & 4 & 8 & 11 & 10 & 5 & 3 & 0 \end{bmatrix}$$

Using this cost matrix the problem can be solved using any of the methods presented in section 3.1. Using an insertion heuristic the quality of the solution may depend on how a node is inserted when it is in the vicinity of the depot, this will be discussed further in chapter 6.

Clarke & Wright algorithm

This subsection presents an algorithm to solve the VRP introduced by Clarke & Wright (CW) in [10]. It is an iterative method, using one initial tour per node to the depot, and it runs like this ([8]):

Step 0: Construct an initial route (0,i,0) for each node i.

Step 1: Create a savings matrix s, where $s_{ij} = c_{0i} + c_{ij} - c_{ij}$ is the savings of constructing a new tour (0,i,j,0) to replace (0,i,0) and (0,j,0). Sort the savings in descending order.

Step 2: Choose the first combination (k,l) in the list if $s_{kl} > 0$. If no positive s_{ij} exists for any (i,j) then terminate.

Step 3 If the new route is feasible combine the routes, otherwise remove (k,l) from the list. Go to step 2.

In step 2, there are two alternative approaches to inserting nodes into the tours, one parallel and one sequential method. The parallel method simply introduces nodes from the list one by one, constructing several tours in parallel. The sequential method starts with the first node in the list, then builds on this subtour while it is feasible (i.e. the capacity of the traveller is not exceeded by adding a node to its tour).

If we consider our sample problem, and add a demand of 3 units at each node and a capacity of 10 units for each of the two trucks, then combined

with the cost matrix from the previous subsection (but without the rows for the virtual depots), the following savings matrix s_{ij} may be constructed[6]:

$$\begin{bmatrix} 0 & 7 & 3 & -2 & -1 & 0 \\ 7 & 0 & 7 & -1 & 0 & -1 \\ 3 & 7 & 0 & 0 & -1 & -2 \\ -2 & -1 & 0 & 0 & 7 & 3 \\ -1 & 0 & -1 & 7 & 0 & 7 \\ 0 & -1 & -2 & 3 & 7 & 0 \end{bmatrix}$$

Using this matrix, the problem and its solution can be seen in figure 3.9. In this case, the solution will be the same using both techniques in step 2, but this is not generally the case. Results from [11] on the accuracy of the Clarke

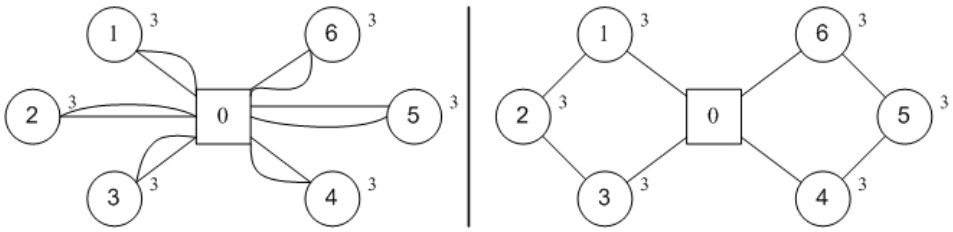

Figure 3.9: Clarke & Wright algorithm. Initiation (left) and solution (right).

& Wright algorithm showed an average deviation of 6.71% above the optimal solution for selected sets of data with known topology and optimum solution.

Fisher & Jaikumar algorithm

Another algorithm for solving the VRP, which may use parts of the Clarke & Wright algorithm, was presented by Fisher & Jaikumar in [12] in 1981. The algorithm uses clustering and savings computations to construct the routes, and works as follows ([8]):

Phase I

Step 0: Choose m initial customers to form clusters with one vehicle each.

Step 1: Compute an insertion cost d_{ik} relative to the initial customers for all customers and clusters.

[6]The depot is not needed in the savings matrix, since all nodes are initially connected to the depot and the savings matrix includes all depot costs through calculation.

Step 2: Solve a generalized assignment problem (GAP) with objective min $\sum_{i=1}^{I}\sum_{k=1}^{K} d_{ik}x_{ik}$, where I is the number of nodes and K is the number of travellers.

Phase II

Step 3: Solve a TSP for each cluster generated in step 2.

How the intitial nodes are chosen is up to the implementer, and can be done in many ways. The same goes for calculating the savings matrix. In the latter case, one possibility is using the savings matrix s_{ij} introduced in the Clarke & Wright algorithm, but with the premise that the tours to the chosen initial customers are included first.

Sweep algortihm

The Sweep algorithm bases itself on placing nodes in tours based on their angular distance from the depot and a chosen initial node. Several approaches has been described for this algorithm, see i.e. [13]. The algorithm works as follows ([8]):

Phase I

Step 0: Choose an initial node, and sort all other nodes after polar coordinates related to this node.

Step 1: Take a new vehicle and include nodes in its route after increasing angle Θ_i as long as the vehicle's capacity is not exeeded.

Step 2: If all nodes are included in a route, go to phase II. Else, go to step 1.

Phase II

Step 3: Solve a TSP for each route generated in phase I.

The choice of starting node could greatly affect the value of the solution obtained. In our sample problem, with characteristics as introduced in the presentation of the Clarke & Wright algorithm, choosing starting node as 1 or 3 will lead to very different results. This is showed in figure 3.10. The difference in the total distance travelled by the two vehicles with node 1 and 3 as initial nodes are 42 and 28 respectively, an additional distance of as much as 50% in the former case.

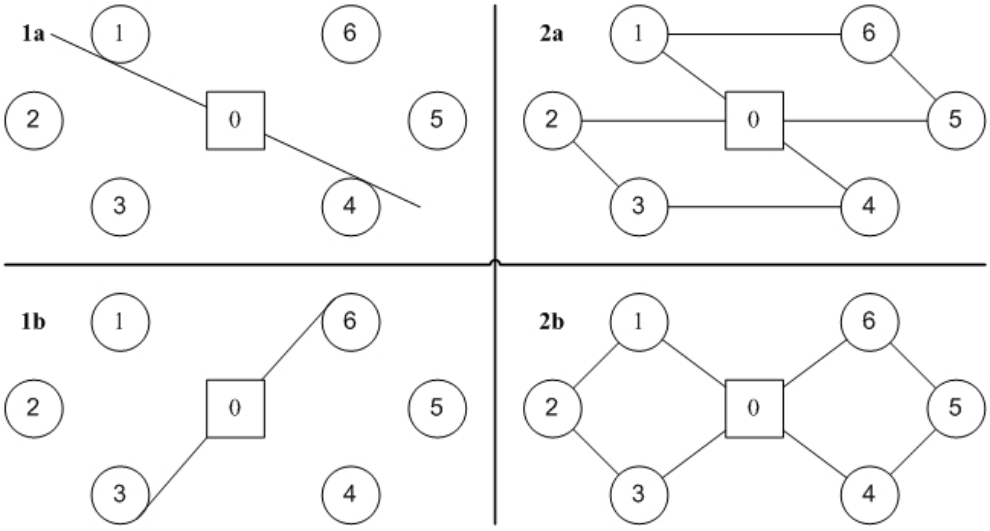

Figure 3.10: Sweep algorithm using node 1(1a) and 3(1b) as initial nodes, with solutions 2a and 2b respectively.

Results from [11] on the accuracy of the Sweep algorithm showed an average deviation of 7.09% above the optimal solution for selected sets of data with known topology and optimum solution.

Other appraoches

Other approaches to effectively solving the VRP in acceptable time for realisticly sized problem exist in a plethora, and algorithms such as Clarke & Wright are still widely studied and enhancements are regularly proposed. While some modern methods, such as tabu search, outperform the "classical" algorithms presented here, they are lacking in one important ability; performance. Since the proposed system will work on daily updated data sets speed is regarded as important, and the modern methods will not be presented here. (It is important to be aware of their existance though, as they may fit other VRP scenarios very well.) For more information on VRP heuristics and meta-heuristics, see [11].

Chapter 4

Localisation Theory

This chapter introduces theory behind an important component of a fleet management system; localisation. At the moment there is one dominating technology for localisation services, the Global Positioning System (GPS). The first section of this chapter presents the theory used in this technology, and how the system is made up. The second section shows how the system may be effectively used in practice, focusing mainly on Geographical Information Systems (GISes). The last section of this chapter will introduce some alternatives to the GPS, one of them interesting for the future.

4.1 GPS

This section introduces GPS. If not otherwise mentioned, this section is based on [14].

4.1.1 History

GPS was made to replace the TRANSIT system, both developed by the United States Military. It deals with two important short comings of its predecessor: The large gaps in coverage (due to as much as 90 minutes between each passing of a satellite at a given position). And the relatively low navigation accuracy on offer. The TRANSIT system consisted of only six satellites, and this is to few to reach the absolute goal of a positioning system: To provide an accurate position at any time anywhere on the planet. The proposed GPS solution which ended up being constructed consists of 21 satellites placed in 12-hour orbits inclined 55° to the equatorial plane. This solution ensures that one has four satellites in good geometrical position at any time anywhere, and was chosen because it completes this task in the

least costly way.

4.1.2 Basic concept

The basic function of a satellite positioning system can be seen in figure 4.1, and what should be noted is: The receiving point consists of three variables; the longitude, the latitude, and the elevation from the earths sea level. Thus, To be able to offer the basic service of such a system, namely providing these three pieces of information, three satellites in range are needed. The

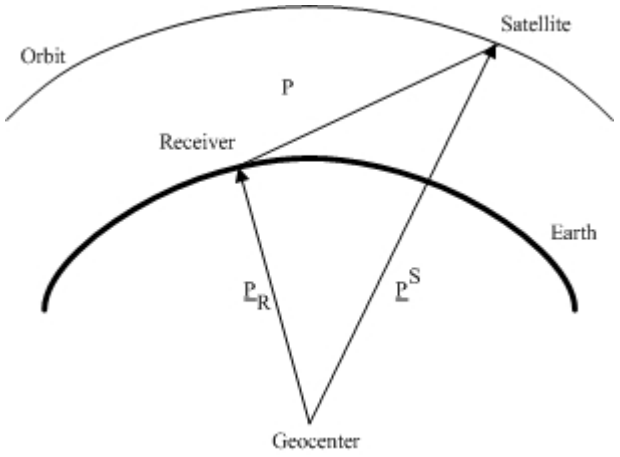

Figure 4.1: Principle of satellite positioning. [14]

reasoning is simple mathematics; if we have three variables, we need three equations to solve the problem. In this case these equations will be

$$\varrho = \|\underline{\varrho}^S - \underline{\varrho}_R\| \tag{4.1}$$

To gain good results using this technique we are dependent of GPS time accurate clocks, and such clocks are expensive. GPS instead uses inexpensive clocks set to approximately to GPS time. Using this technique the estimated range will be slightly longer or shorter than the exact range. This estimated range, called the pseudorange, can be expressed as:

$$R = \varrho + \Delta\varrho = q + c\delta \tag{4.2}$$

Using this technique we now have four variables, the pseudorange and three position components related to equation 4.1. This is solved in GPS by using the signal from four satellites to obtain the exact position. It can be concluded that the accuracy of the determined position hinges on the following factors:

- Accuracy of each satellite position.
- Accuracy of pseudorange measurements.
- Geometry.

To deal with systematic errors in both satellite position and pseudorange clock biases, the use of an interferometric technique can decrease and even remove these errors by differencing the measures from two sites to the satellite. Poor geometry though, cannot be overcome by this technique or any other differencing method. To measure a satellite's geometry with regards to the receiver site is known as the Geometric Dilution of Precision (GDOP) factor, but further discussion of this technique is considered outside the scope of this report.

Determining the velocity of a moving vehicle is another goal for navigation, and thus the GPS. Since a satellite has relative motion to a moving vehicle, the frequency of the broadcast signal is changed when received by a vehicle. This is called a Doppler shift and is both measurable and proportional to the relative radial velocity. The satellite's radial velocity is both constant and known, thus the radial velocity of the vehicle can be deduced from the Doppler shift.

Surveying

Based on the theory presented thus far in this section several extensions to permit more advanced GPS functionality have emerged. One such functionality is GSP surveying. By using the interferometric technique presented in 4.1.2, it is possible to use GPS for geodetic surveying. Geodetic surveying allows us to determine the length of a vector between two points, meaning the vector length between two GPS receivers, with millimeter accuracy. This technique is called relative positioning. The accuracy is dependent of the distance between the two points, and it decreases as the distance increases. This type of functionality is useful not only for point to point navigation, but also for developing detailed maps of desired areas. For a detailed presentation of how GPS surveying works, see [15].

There are several observation techniques related to relative positioning. In most early day surveying both receivers where placed at fixed points, a technique called static surveying. Following a rapid decrease in time needed to complete a survey, new methods such as kinematic surveying emerged. In kinematic surveying one of the receivers moves while the other remains fixed,

and already in the mid-80s sub-centimetre accuracy was achieved in a few seconds using this technique.

In addition to relative positioning, a technique called differential positioning is used today to achieve accurate positioning in real-time. This technique is based on having a receiver at a known, fixed point, and letting this receiver regularly transmit updated pseudorange corrections to mobile receivers within its operative area in a closed interval.

Signalling

A GPS satellite broadcasts two types of pseudorandom noise code on two carriers. The first type of code, called the C/A-code (Coarse/Acquisition-code), is modulated on only one of these carriers and omitted from the other one, allowing the operator, JPO (Joint Program Office), to control the broadcast information. It is this C/A-code based positioning, called SPS (Standard Positioning Service), that is available for civilians. The other code, the P-code (Precision-code), is used by the U.S. military and other authorized users. The P-code is sent on both carriers, and allows for higher accuracy than the C/A-code. In addition to these two codes, a data message containing status information, satellite clock bias, etc. is modulated onto the carriers.

4.1.3 User groups

As mentioned in the signalling part above, there are two main user groups related to GPS, civilian and millitary.

Military users

GPS is made and controlled by the United States military, and is used for the following purposes:

- Coordination. The idea from the start of development was equipping every military unit, on land, sea or airborne, with a GPS receiver allowing for advanced coordination and cooperation between both intra- and inter-branch units.
- Unit control functions. By connecting a receiver to four antennas on the same unit useful information about i.e. route derogation can be collected.

Civilian users

For civilian users, GPS is mainly used for the purposes listed below:

- Navigation. The first plans for civilian GPS usage was for simple navigation services.
- Surveying. The interferometric concept used in GPS make it usable for detailed, short line, land survey measurements.
- Fleet management. Fleet management systems such as the one being proposed in this Master's Thesis are more and more common, and use GPS for positioning and location. (More on GPS in fleet management in chapters 5, 6 and 8.)
- Emergency services. Combining GPS receivers with map systems in response vehicles enables effective routing of these vehicles in areas carrying a great deal of traffic.
- Various utilities for residentials. Technology ranging from simple GPS receivers for use with hiking and tracking, to relatively advanced solutions for i.e. driving assistance already exist in abundance.

Since most of the status messages available to military users are unavailable to the general public, several services have been developed to offer an acceptable amount of system information to civilian users as well.

4.2 GIS

This section will present Geographical Information Systems (GISes). These systems use geographical information, often in connection with positional data, to offer advanced services. Such a system allows linking attributes to location data, and can be used in many ways. A fleet management system such as the one proposed in this text could use GIS to obtain the distances between all jobs, offer an overview of all trucks in the desired area and their routes to the central office, and offer driving instructions to each truck. All these tasks are easily within reach for a well constructed GIS solution.

4.2.1 GIS views

While maps are the obvious first thing to mind when talking about GIS, [16] separates GIS into three possible views:

1. Database view. GIS is a geodatabase; a database describing the world in geographic terms.

2. Map view. GIS offers geovisualisation by allowing the use of intelligent maps of earthly features to support queries, analysis, etc.

3. Model view. GIS is a set of tools offering geoprocessing by applying functions to existing datasats, thereby creating new datasets.

Below, in-depth information on the different views are presented.

Database view

GIS geodatabases consists of features, and these features are usually connected with a given way of presentation. For example a street is mapped as a centerline. The features are collected into feature classes, and are used to create data sets geographically presenting som aspect of the world. There are lots of such data sets, including ordered collections of vector-based features, raster data sets, networks, terrains, survey data sets, and many others. For an example of an address data set, see figure 4.2. To offer complete repre-

Figure 4.2: An address data set. [17]

sentation of the data available, a geodatabase needs to obtain standard tabular attributes describing the geographical objects. Other important parts of a GIS database is spatial relationships, such as topologies and networks. Topologies is used to i.e. manage feature boundaries and apply integrity rules, while networks are descriptions of GIS graphs which can be traversed. The geographic data in a GIS database are organized as thematic layers, and, since they represent real-world locations they overlap each other. An example of such a layered data set can be seen in figure 4.3. This section was based on [17].

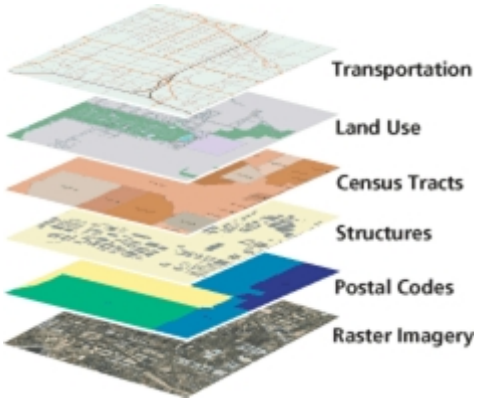

Figure 4.3: GIS layered data set. [17]

Map view

Map view does not only comprise maps, but also other visualisations such as charts, tables and schemes. A more fitting name is thus geovisualisation view. All these views are used in GIS to provide an interface between users and the geographical data sets, allowing users to use the existing data interactively for advanced data compilation, cartography, analysis, etc. This section is based on [18].

GIS maps come in various forms; simple versions are often used in i.e. small handheld devices, while advanced business solutions often use advanced maps as well. Interactive as these maps are, they have the following properties:

- Pan and zoom. While using these functions different map layers will turn themselves on and off appropriately based on current map scale.

- Apply symbols based on attributes. The symbol for a city may for example be based upon its population, and these possibilites allow for more informative maps.

- Use of pointers. In interactive maps such as those in GIS, it is possible to point at map objects to obtain furher information about the object, to perform analysis on the object, etc.

Basically, maps are the most used interface to users of GIS, and supply GIS data set information in an approachable way.

Model view

Creating new data sets are another important function of GIS, and is based on using operators, called tools, on geographical data sets. A GIS contain a lot of such tools (possibly thousands), which allows for advanced geoprocessing. A simple example of such processing can be seen in figure 4.4, and by using different operators sequentially much more advanced processes than the one in this example may be carried out. Geoprocessing is used with almost any

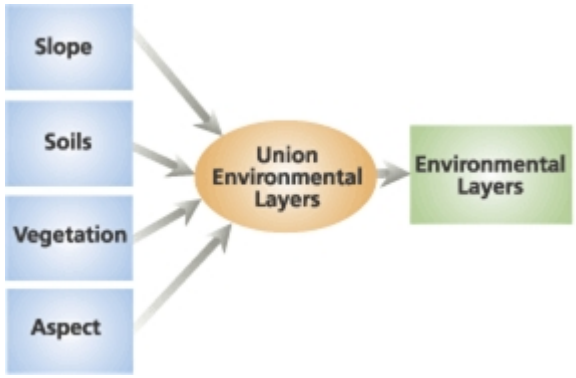

Figure 4.4: A simple GIS model. [19]

task in a GIS, and often in collaboration with the other GIS views. This section was based on [19].

4.2.2 GIS implementation

There are several considerations to take into account when developing a GIS, and this section will present some of the existing challenges. In [20] the author, Tomlinson, proposes the following ten steps to successfully develop your own GIS solutions:

1. Consider strategic purpose. The first thing that must be clear when one develops a GIS is its purpose. What functions the GIS will offer, and how it benefits the company, is what everything else is built around, and a thorough work on this point ensures that the company gets what it needs to fullfill its strategic plans.

2. Plan for planning. Before any implementation may take place, thorough planning is a necessity. The result of the planning process will be a project proposal, and if this proposal is approved, and the necessary resources are committed, then the chance of producing a successful GIS is already high.

3. Determine technology requirements. In association with the client the GIS' requirements must now be defined. This is done by realising what inputs and outputs the system must provide, thus revealing the actual user perspective requirements.

4. Determine end products. The actual output of the system, described as information products, is determined in this face. To achieve this, one must survey what the users' jobs are, and how the system should be of help to them. The result of this stage should be a complete view on what information products are needed, how often they are to be created, which error tolerance each of them have, and what benefits they bring. What functions are needed to produce these information products should also be included in the report from this phase.

5. Define system scope. When the previous phases are completed, the next step is determining then scope of the entire system. In this phase what input is needed, and when, will be determined, along with which information product(s) each input is used to produce.

6. Create data design. In this face the actual conceptual design of the GIS is created. To do this successfully, collaboration and iterative database design are among the important factors.

7. Choose logical data model. A data model describing the parts of the real world which are interesting for the company is needed. This model is used to effectively retrieve the data necessary to carry out the complete system's tasks.

8. Determine system requirements. In this phace necessary system functions and user interfaces are examined, along with interface, communications, software and hardware requirements. The result should be an overview of all these needs, and also propose hardware and software configurations.

9. Cost and benefit analysis. After completing the first eight phases, the conceptual design is complete. It is now time to focus on how to most effectively implement the system, and this may include a cost-benefit analysis and a business case for the system. Surround matters, such as law issues and staff training, should also be handled in this phase. The result of this phase should be a complete implementation plan.

10. Make implementation plan. The last phase comprises making a complete implementation plan, consisting of all the information gathered

through the previous stages. This final plan should work as a guide for the actual implementation, leading to a, hopefully, successful end result.

4.3 GPS alternatives

In this section two alternatives to GPS in the context of positioning are presented. First, the new European positioning system, Galileo, will be presented in section 4.3.1. Second, mobile network positioning will be looked into in section 4.3.2.

4.3.1 Galileo

Galileo is a new global navigation system, under development by ESA (European Space Agency). Unlike GPS, Galileo is a civilian system, and all its services are meant to be available to the general public. While it is specified to be independent, Galileo will be interoperable with services such as GPS. The accuracy requirements for Galileo with respect to some different aspects can be seen in table 4.1, notice the differentiation of the market and the security segment. "Mass market" in the table refers to i.e. vehicle and mobile

Primary Application	**Mass Market**	**Safety Related**
User Masking Angle	25°	5°
Accuracy (95% Confidence)	10 metres horizontal	4 metres vertical
Coverage	Global	
Availability	Better than 70%	Better than 99%
Integrity	Not generally required	Mandatory

Table 4.1: Navigation Service Requirements. [21]

receivers in city areas with limited sky view, and the availability is as low as 70% because users are not deemed to be in need of the services continuously. "Safety related" is directed at air and sea units, or others with a good sky view.

Satellites

When Galileo is complete it will have a constellation of 30 satellites, 3 of which are operative reserves. These satellites will circle the earth with a 56° angle with respect to the equatorial line, at an altitude of around 23 thousand kilomtetres ([22]). The satellites are to be placed in an optimal constellation, and together with the availability of spare satellites already in orbit a failure for one satellite should not undermine the service provided significantly. The satellites will be placed in three planes of ten satellites, and each satellite will use approximately 14 hour to circle the earth. Figure 4.5 show how the satellites will be placed. Most places, the Galileo constellation plan should

Figure 4.5: **Galileo satellite orbits. [23]**

result in six to eight satellites being visible at any time, allowing for navigation accuracy at centimetre levels. Even in cities where signals may be blocked by the surroundings, the chance that a user will have the required number of satellites available for positioning is high.

For the service to be working properly, it is naturally important that the satellites are placed exactly where they are meant to be according to plan. The Galileo satellites will have the technology necessary to stay in place when already in position, but they will not have the technology required (engines) to obtain this position themselves. This job needs to be done by a separate launcher. Because of the delicacy of this process, ESA will undertake this operation stagewise. (All this information is gathered from [23].) In the first stage, a test satellite will be launched, and placed in the first orbital

plane. The test period will last two and a half years, and will be used to test satellite equipment such as the atomic clocks, as well as ground station functions. When the test period is over the first four operational satellites will be launched, two into the first orbital plane, the other two into the second orbital plane. Using advanced system simulators, these four satellites and some of the ground functionality will be used to validate the Galileo system. Following validation, the last stage of the Galileo launch program will take care of launching all the residual satellites as quickly as possibility, thus providing the service to regular users.

Ground equipment

At the ground, Galileo will consist of the following parts ([22]):

- Twenty Galileo Sensor Stations. These stations, placed around the globe, will receive signalling from the satellites, and transfer it to the Galileo Control Centres.

- Two Galileo Control Centres. Both placed in Europe, the control centres will offer satellite control and management. Among their tasks are compute integrity and synchronizing satellite time signals.

- Fifteen up-link stations. Placed around the globe, these stations are responsible for sending information from the control centres to the satellites.

Status and development plans

The test satellite, called GIOVE-A, was launched December 28^{th}, 2005, from Baikonur in Kazakhstan ([24]). The test phase will last until 2007, including the launch of the 4 first operative satellites. During 2008, the plan is to launch the remaining 26 satellites, thus being able to offer services to the general public.

4.3.2 Mobile network positioning

Unlike the two dedicated positioning systems presented in this section, GPS and Galileo, mobile networks such as GSM and UMTS can offer positioning that is not satellite based. Several techniques for such calculations exist, and some of them will be presented in this section.

Cell-ID

The simplest form of dealing with positioning in mobile networks is Cell-ID. In this technique the base station to which the ME (Mobile Entity) is currently connected, and identify the place where the base station is located as the ME's location. This type of positioning is inaccurate, with accuracy in the range of 500 metres to 20 kilometres in GSM and 100 metres to 5 kilometres in UMTS, depending on the size of the cell the base station is covering ([25]).

E-OTD and OTDOA

A technique frequently used for retrieving an ME's position in GSM is called E-OTD (Enhanced Observed Time Difference). This technique uses signals both from the ME and a fixed receiver at a known point, and calculates the instantaneous transmission time offsets of each transmitter relative to its neighbours at the fixed receiver This can be done due to the fact that the fixed position of all transmitters are known. Timing offsets measured by the ME can then be used for calculation, where the position of the ME is predicted by the points of intersection of two or more hyperbolic position lines. To work, E-OTD needs to measure the signals from at least three transmitters, and these must be at geographically distinct places. E-OTD has been made a mandatory part of UMTS, where it has been renamed OTDOA (Observed Time Difference of Arrival). (This section has been based on [26] up until now.) E-OTD offers better accuracy than Cell-ID, 100 to 500 metres according to [25], but also leads to larger amounts of signalling and extra network components (the stationary receivers) than Cell-ID does. (The accuracy, and drawbacks are the same for OTDOA in UMTS networks.)

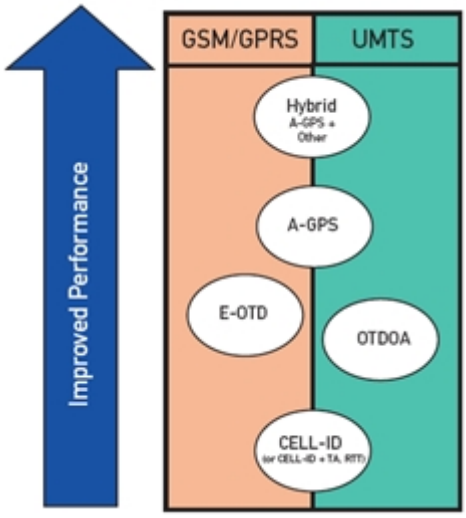

Figure 4.6: Mobile network location method accuracy. [25]

What should be noted is that none of these methods really offer good enough accuracy for location based services. To obtain this in mobile networks, a GPS unit is most likely necessary. According to [25] a so-called mobile A-GPS (Assisted GPS) solution can offer accuracy typically in the range of 5 to 50 metres, but then again this is a GPS solution and not a pure mobile network solution. For a visual representation of the accuracy of the different technologies, see figure 4.6.

Chapter 5

Design

This chapter will present the proposed system design for the fleet management software this Master's Thesis aims to develop. While the scenario and some of its challenges were presented in chapter 2, this chapter will offer a much more thourough presentation of the challenges in developing the system. In addition a complete requirement specification for the problem, and UML (Unified Modelling Language) diagrams will be presented to visualise different aspects of the proposition. While the system seen from a user's perspective will consist of two applications, one for the central office and one for each field agent (truck), these two applications will be derived into several packages each in this presentation. This is done because the both applications must perform a variety of tasks, and the processes involved in each task is seen more easily using decomposition. The requirements for each package will be presented and discussed in section5.2 and modelled in detail in section 5.3. A test plan is then presented in section 5.4. But first a detailed, top level requirements specification will be derived in section 5.1.

5.1 Requirements

In this section a requirement specification for the proposed system will be made. To do this, let us first look at the requirements that the users will have, and use these to develop a more technically specific plan thereafter. The users can be divided in three groups: Those working with the central application (operators), those working with the field application (drivers), and externals. The two first groups are seen as the most important in this case, but since functionality which favours both customers and especially management will most likely be required in a commercial system, these groups' needs will be (at least partly) assessed as well.

Operator requirements

The operators will esentially manage the hub in the system. All information from both field agents and customers end up in the central office and consequentially all tools necessary for appropriate processing must be located there. To keep the system up to date this means that the operators must take care of the following tasks:

1. Ensure that the system contains correct information about all system entities (drivers and trucks) at all times.
2. Ensure that jobs are added or removed based on customer request.
3. Offer support information to customers on request.
4. Have visual control of each truck.
5. Communicate with drivers at need.
6. Initiate the optimisation process.

To fullfill the first point in the list above, the system must offer options for adding, removing and updating driver and truck status. To deal with adding and removing drivers and trucks, it is natural to provide a GUI (Graphical User Interface) for these operations. In the case of updating information, it depends on what type of information is involved. If it is an update of unit specific characteristics, such as i.e. driver address or truck abilities, it should be adjustable the same way as in the case of adding or removing. If the information concerns the connection between units, such as a driver and a truck, the update process should be automated in the system. When a driver is assigned to a truck, this information should be handled by the system, without the need for operator interaction. (This will be discussed further in the section about driver requirements.)

The second point in the list would naturally be handled in the same way as with trucks and drivers, using a GUI. How new jobs may be registered by the operators before being added to the system may vary. The simplest way would be by telephone, but having i.e. a scripted website allowing online job registration is certainly a possibility.

In a well working fleet management system there should be enough transparency to allow operators to give requesting customers a good estimate as to when they can expect the job they have registered to be carried out. (This point (number three in the list above) has not been among the major driving

forces behind the design being presented here, but should be included in any commercially available system.)

When it comes to visual control of each truck (point 4), there are lots of possibilites. The best way to ensure this will be to include a GIS in the system. This allows the operator not only to see the trucks placed on a map, but also to use the map interactively to obtain desired information. Such a system would simplify the operators' job, as it lets them easily access the different trucks in a point and click environment instead of through more teadious search methods. (Such easy accessibility would also simplify task 3 in the list.)

There are two natural ways that the central office would like to use when communicating with the field agents: Voice and data. While the main operation of the system should be based on data transfer between the field and office application, which is a process that should be largely automated, voice communication should be available as personal communication may be necessary in some situations. Whether voice communication should be integrated into the system itself, or be a parallel service, needs to be decided. In this proposed solution it will be set aside, mainly because it will mostly be needed in case of problems, and if there is a system error voice communication should still be available.

The optimisation process is the last part the operators need to take care of. It will usually be done in the beginning of the day, and includes all jobs registered before a given time. It is this process initiation that creates the job lists for each truck, and thus driver, in the beginning of the day. However, as jobs may be added during the day the process may have to be run over, this will be further discussed in section 5.2.

Driver requirements

The drivers are responsible for carrying out all the jobs the company has obtained, and most of these tasks are carried out without any interaction with the system. For the complete system to work however, it is important that the field application takes care of the following tasks:

1. Assign driver to truck.

2. Blocks any action (except for driver login) before a driver is actually logged in.

3. Does not allow multiple simultaneous logins.

4. Enable updating of job status.

5. Offer information on current joblist.

6. Offer driving instructions.

To assign a driver to a truck, a login mechanism of some sort should take place at system start-up. This could be done by a simple username-password mechanism, or by f.i. a card solution using PIN codes. The first solution can be software based and thus cheap, however a good solution as to how passwords should be treated is needed. (Having them unencrypted in the database certainly makes them vulnerable. More on this in section 5.2.) The card solution is more costly, since it requires a card reader for each field agent, but it will also be harder to compromise.

For the system to offer good status reports the status of each job must be updated correctly. To do this the driver needs to update the status of each current job accordingly. This way, the central system, and thus it operators, knows which jobs are completed, which are currently executed and which are not yet handled. This information can be useful especially when assigning new jobs during the day, when the actual status of each joblist will give better insight than the static estimates made in the morning.

Providing information on the current joblist offers valuable information to the driver, and may allow him or her to better plan his days. While it may not be the most efficiency increasing part of the system, it is still a natural part as such insight will most likely help in gaining driver acceptance of the system. If a driver had no overview of his day's work, he could never accept the system.

Driving instructions, typically via a GIS, may in many cases be unnecessary for experienced drivers. However such information could be helpful in situations where drivers cover new areas, or if roads are closed. The GIS in the field agent could be much simpler, as no interaction should be necessary. (See section 5.2.)

Requirements from other actors.

In addition to operators and drivers there are two other actor types connected to the system, these are the customers and the company management. For

customers the requirements are closely connected to some of the points made for the operators. Customers basically may direct three requests to the system: Add job, remove job and retreive job status. The simplest way to allow for this is to simply let operators handle these requests over the phone. If this solution is used, the customers do not have any direct contact with the system and thus are not directly affected by its operation. In the case a web solution is used, allowing customers to add jobs directly into the system themselves, they become an active part of the system environment as well. The subject of customer interaction will not be discussed in greater depth in this report, but is mentioned because it needs to be clarified in a system meant to be used commercially.

An increased amount of information available to management is a natural aspect of implementing a fleet management system in your company. If statistics are saved at the end of each day, including both estimates made the system and factual performance data, the following information could be available to management:

- System reliability. The estimates made by the system can be compared to actual time used on the different routes to assess the quality of the estimations made.
- System efficiency. By comparing the performance of the company before and after the system is introduced savings as a result of the system may be calculated. Efficiency calculation will probably offer most realistic results after a while, as comparisons needs to be made between similar job data sets. (number of jobs, locations etc.)
- Driver performance. Since each driver is "logged into a truck" it should be easy to assess each driver's performance based on truck specific data.

Such management functions are a natural part of a system of this sort. However, it will not receive much attention in this report as it is not considered an important part of the general system functionality.

Requirement specification

Taking into consideration what functions each actor related to the system requires, table 5.1 presents a complete requirement specification for the the whole system. The table gives each requirement a number, relates it to one of the two applications (or both), and gives it a priority. Some of the requirements is resolved directly from the descriptions earlier in section 5.1, while

others follow implicitly.

The requirement specification will be used in sections 5.2 and 5.3 to create a complete model for the system, and also make up the basis for the system test plan found in 5.4. Before moving on to the modelling a few points should be mentioned about the priorities given to the requirements. A high priority is given to all tasks which are considered critical for system performance. Tasks with medium priority offer valuable service and would be natural to include in a complete solution but would not affect the main functionality of the system if non-existant. To give an example: Driver data such as i.e. addresses *should* be correct, but the main functions in the system does not depend on it. Hence medium priority (R7). Truck data may however include facts about work related abilities which may limit what jobs a truck could do, and such data *must* be correct for the system to work correctly. Hence high priority (R12). (In this scenario trucks have no special abilities, so this potential situation will not take place.) Low priority has only been given to management functions and customer interface, as they are considered outside the scope of the proposed solution.

5.2 Model Discussion

It is time to use the requirement specification to evolve a complete system model. As mentioned in the introduction to this chapter the system will be divided into packages, each package used to fullfill some of the requirements. To split the system into functional packages the perfomance the requirements have to be assessed in a more technical manner than they have by now. It is natural to split this section in two; one part for the central application and one for the field application.

5.2.1 Central application

The central application is by far the largest part of the system, and most of the requirements and challenges are related to it. Many of these challenges will be discussed here, to create a solid system model.

Adding and removing units

Adding and removing units, both trucks, drivers and jobs, should be a simple task seen from the operators' view. To achieve this, it is natural to have a GUI for each of this task, accessible from a menu or a button in the central

Req nr	Description	Application	Priority
R1	Add job to system	Central	High
R2	Remove job from system	Central	High
R3	Update job status	Field	High
R4	Retrieve current job status	Central	Medium
R5	Add driver to system	Central	High
R6	Remove driver from system	Central	High
R7	Update driver data	Central	Medium
R8	Assign driver to truck	Field	High
R9	Ensure correct login mechanisms	Field	High
R10	Add truck to system	Central	High
R11	Remove truck from system	Central	High
R12	Update truck data	Central	High
R13	Otimise jobs	Central	High
R14	Offer view of all truck positions	Central	High
R15	Offer driving assistance to drivers	Field	Medium
R16	Offer joblist information to drivers	Field	High
R17	Offer data communication between office and trucks	Both	High
R18	Offer voice communication between office and trucks	Both	Medium
R19	Support management functions	Central	Low
R20	Offer online job registration for customers	Central	Low

Table 5.1: Requirement Specification

system's main window. Since there are several requirements that needs this kind of GUI available, having a visual package containing all the interaction functionality seems an obvious choice. While these functions will be the ones visible from an operator's angle, they are worthless without corresponding functionality beneath. To obtain all information about these units, the natural approach is to introduce a database. The add and remove requirements (R1, R2, R5, R6, R10 and R11), plus some of the update requirements (R7 and R12), imply that the database must contain a tables for jobs, drivers and trucks. While the database itself is a separate unit, it is necessary to have a link between the database and visual parts, and so a separate system core package, to co-ordinate the functionality of other system parts.

Graphical view

The central system should offer operators a map view of the are of operation (In this scenario, the City of Trondheim), showing at least the position of each truck (R14). By creating a GIS to deal with this part of the solution, there are few limits to the functionality which can be made available as a result of the interactivity such a solution may offer. There is however need to consider which functions would fit in a fleet management system, and some of these are:

- Access route by clicking map. By clicking a truck on the map (in the GIS, it would be natural to give trucks their own symbol), it should be possible to retrieve a status report for this unit including information such as an overview of its route, including which jobs it is assigned to and estimated time left, who the driver is, etc. An alternative would be simply marking the assigned route, including all jobs, directly on the map. Independent of the chosen method, such functionality allows operators excellent access to information in a very accessible way.

- Access job info by clicking map. It could be possible to mark all jobs on the map as well, possibly using different symbols based on their current status. This could be very helpful for customer assistance and fullfills requirement R4, especially if a good search method to find the correct job on the map is included.

- Offer distance calculation. A GIS can be used to calculate the time needed to drive between point A and B, this functionality will not only be helpful, but is a necessity in the optimisation process. (More on this when the optimisation requirement is dicsussed a little later in the text.)

While the GIS part could certainly be included as a part of the visual package, due to the complexity and special features it offers, and the fact that is works quite different from the other GUIs it is created as a seperate package.

Optimisation

A main feature of the system is the optimisation part (R13), which often will be the main reason for a company to acquire a fleet management software. The theory behind the optimisation part can be found in chapter 3. To carry out the optimisation, an operator should simply have to push a button in the system's main window, the rest of the process should be handled within the system core. While the theory shows that optimisation in a scenario such as

this can be advanced an time consuming, the solution proposed here bases itself on an heuristic approach which should not be to hard neither computationally nor implementation wise. Including a class in the core system to handle this process should therefore be sufficient. However, for the optimisation process to be accurate, it needs good data from its surroundings. To solve the VRP it needs a cost matrix offering travelling times between all existing jobs. To achieve this, the process depends on the geographical calculation abilities available in the GIS package. The output from the optimisation process will be a set of routes, or joblists, that will also need to be saved in the database. When optimisation happens later in the day, rerunning the process is probably not a good idea. (More on why this is in the disucssion in chapter 8.) This means that a plan for such a case, which is likely to occur, is needed as well.

Communication

The last functionality needed in the central application is the ability to communicate with the field application (R16). This communication should be initiated as a result of some information update, and as such should be dealt with in the core system. (This is based on a solution where voice communication (R17) is dealt with separately.) This communication should be made using secure connections, especially since driver passwords may be sent over these connections.

5.2.2 Field application

The field application has much fewer functions than the central one. It needs the same communications abilities as the central application (though triggered differently), and in addition must take care of the following areas: At system startup the field application must offer a login function, using a GUI login window (R8). In addition it should have the ability to offer two other views: One providing the joblist (R16), the other offering driving instrcutions using its own GIS (R15). It is also important that the field application core ensures that exactly one driver is logged in, if this is not the case all action should be restricted (only login allowed (R9)). The GIS could be made advanced (3D driving instructions are common nowadays in mobile navigation assistants.), or simple (using a 2D view with arrows pointing the direction). To be able to offer this, the field application needs functionality to communicate with a position provider, such as i.e. a GPS receiver. Apart from this, the only functionality needed is the ability to change the status of a current job (R3), and this could be done using only a button, or a button and a

dropdown list. The core of the field application has one more very important task; to regularly send the position of the truck (based on GPS data) to the central agent. This is necessary if the central application are to display an accurate view of all trucks' current positions.

Since there is a lot less functionality in the field application than the central application it will be modeled as just two packages, one for the core functions (including basic visual functions), and one for the GIS solution.

5.3 Model presentation

Based on the previous two sections it is time present a top level model of the system, and it can be seen in figure 5.1. To add some insight lets look into each package and sum up its necessary functionality.

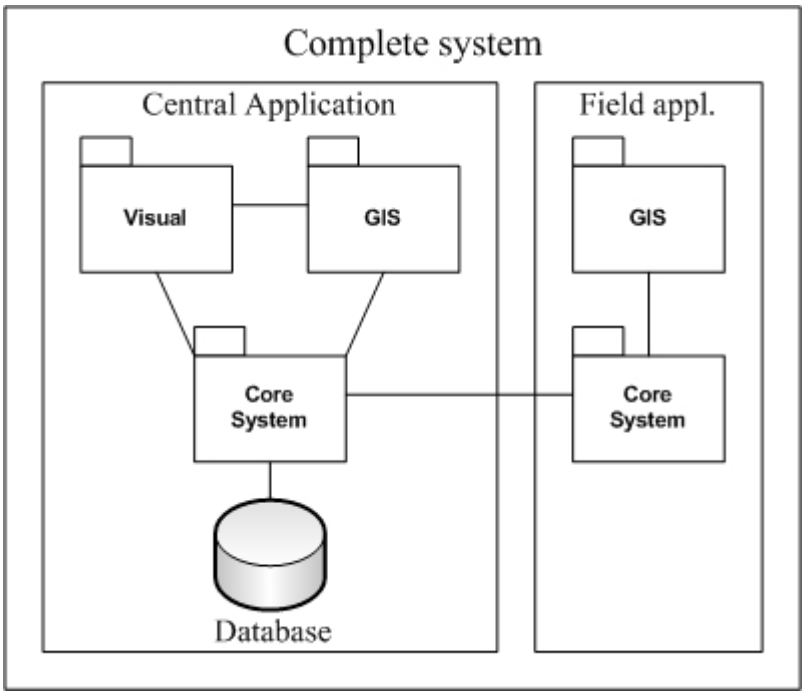

Figure 5.1: A top level model of the complete system.

5.3.1 Central application GIS

As previously mentioned, the GIS in the central application is meant to offer graphical views and services to a operators. Its main function is to simplify

operator jobs, and to assist the system optimisation process. To give a more detailed specification of the GIS design, let us use the design approach presented in section 4.2.2. Results are summarised in table 5.2.

Question	**Answer(s)**
What is the purpose of the GIS?	Assist optimisation process. Offer easy access to system data for operators.
What are the inputs and outputs?	Inputs: (1) New job address, (2) Map click request. Outputs: (1) Distance vector from new job to all existing jobs, (2) Map click reply.
What are the end products?	Interactive city map with field agent positions, updated real-time.
What is the scope for the complete system?	Offer fleet management to the company.

Table 5.2: Central application GIS essentials.

This gives an overview of some important specifications for this GIS. It is however far from detailed enough to work as a basis for the implementation. A detailed database design for the GIS will not be presented in this text[1]. To be able to work as desired however, it can be determined that the GIS will need to contain the following data and functionality:

- A vast variety of map data, in this case covering the City of Trondheim. The GIS database needs to contain all map layers necessary to perform travel time estimates between points. This means that maps containing detailed road info such as speed limits, driving directions, etc are available. Without a well working GIS these tasks can not be performed, and it will render the complete system rather useless. Such maps are usually published by some branch of the authorities, in Norway basic map data sets are administrated by Statens Kartverk ([27]), and detailed road information is offered by Statens Vegvesen ([28]).

- Map visualisation. To offer operators the possibilities presented earlier in this chapter the maps most be built and shown graphically. The map should be interactive, and entity specific symbols (such as a truck symbol) should be available. To create these maps the GIS will need the tools to create a layered map. As an example: Put the basic land

[1]Since the GIS part will not be implemented the author finds that a higher level description of necessary functionality is adequate.

map in the bottom, then add maps for roads, houses etc on top of this, and add symbols etc on the top level based on interaction.

5.3.2 Visual

The visual package will contain all functionality necessary for the operator to do maintenance and optimisation tasks. By offering the main system window, and all necessary dialogs, the visual package's simple goal is to graphically offer the operator the desired functionality in the desired way as simple and effective as possible. The main window will typically be connected with the system core for system internal communication, and also sets the premises as to how the GIS is presented to the operators. Since the system will have quite a few functions, detailed design information, in the form of an UML class diagram, has been placed in appendix B.

5.3.3 Core system (Central application)

The core system in the central application is in many ways the internal system hub. All communication between other packages (with the exception of the GUI-GIS connection) goes via the system core. Its responsibilities are:

- Communicate with the database.
- Communicate with the visual package.
- Communicate with the field agents.
- Take care of the optimisation process, via calculations and communication with the GIS.

Based on these requirements, a proposition for required classes within the system core can be seen in figure 5.2. The class DatabaseHandler atakes care

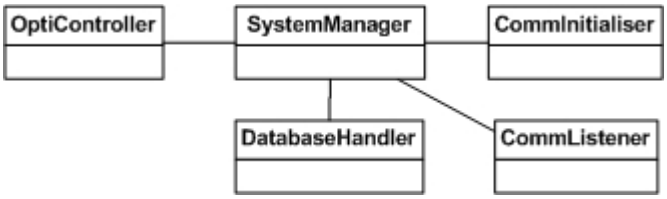

Figure 5.2: Design for package CoreSys.

of database connections, while CommInitiator and CommListener together offers communication with the field agents. The OptiController contains all

optimisation methods, and execute these when requested. The main entity in the system core package (called CoreSys) is the SystemManager class. This class is the control class of the system, and is responsible for all inter-class and -package communication and the correctness of the system's behaviour. For a complete class diagram including attributes and methods, please refer to appendix B.

5.3.4 Database

The system database is necessary to keep track of the status of all the different units related to the system. It is important to note that this database does NOT include GIS data, both GIS packages will have their own, internal database. Figure 5.3 shows how the database has been designed, suiting all the system requirements.

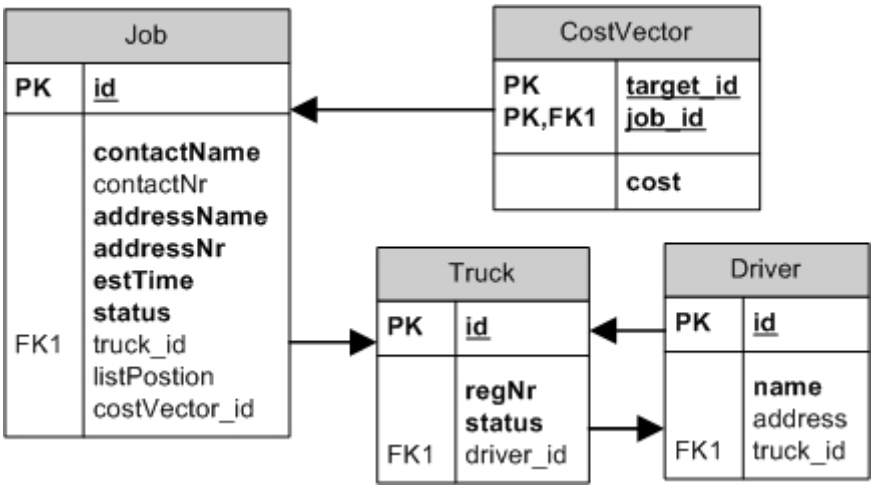

Figure 5.3: Database design.

This database model should take care of all the tasks it has to based on the requirement specification. The capabilities of the database if implemented based on this design is:

- Handle all characteristics necessary for the three system entities.
- Associate a driver with a truck (and vice versa).
- Associate a job with a truck, and assign it to a position in the truck's route.

- Keep an up to date status for each truck[2].
- Keep an up to date status for each job.
- Contain a cost vector for each job to all other jobs, enabling the creation of a complete cost matrix.

Note: The dark fields in figure 5.3 is fields that cannot be empty (fields with the ""not null" tag). The other fields may be empty, but if the truck_id field in a job is set, meaning that it now features in a route, the field listPosition must have a value as well. This can be realised in two way: Either it can be added as a constraint in the database, or it can simply be ensured in the methods used for updating the database.

5.3.5 Field application GIS

For the GIS in the field application the purpose is: Based on current position and current target location, offer efficient driving assistance to the driver. This means that the input to the GIS in this case will be position data from the GPS receiver, and that the output will be an on-screen route from current location to desired location. Such functionality does require most of the map functions used in the central application GIS, though it does not need to offer any direct interaction. To create a more advanced service, the maps used in this GIS may very well be three dimensional. As with the central application GIS no detailed design will be offered for this GIS neither.

5.3.6 Field application system core

The field agent core contains the window used to offer information to the driver. This window will contain three views, a view of the current list of jobs, which can be realised internally in the window class, a map view, provided by a GIS, and a system view for login functionality. Communication works like in the central application, and the SystemManager also has a similiar place in operation, but with different tasks. The model for the field application can be seen in figure 5.4, and also exists in an extended version in appendix B. Short explanation: MainWindow contains the three views, and thus takes care of GIS interaction. LoginWindow ensures that when the application starts, a driver is associated with it. InfoHandler takes care of all communication with the central system, and CoreSys is the control class in the field application and manages position data as well.

[2]This relates to the connection status between the field agent in a given truck and the central application.

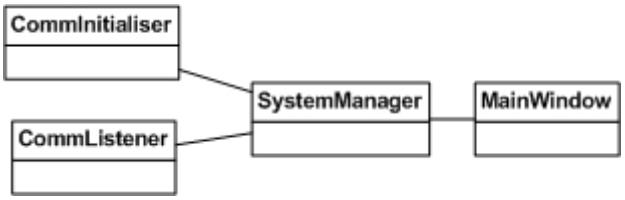

Figure 5.4: Design of package FieldCore.

5.4 Test plan

Before moving on to implementing, a test plan should be in place for the system. The test plan can be found in tables 5.3 and 5.4. The first table contains numbered test entries, which requirement they relate to (numbering from table 5.1), and a test description. The second tables contains information on how each test should be executed, and a result colunm (I/S column[3]). The results will naturally not be added to the table until after the implementation. (The reasoning behind using two tables instead of one strictly relates to layout issues.) It should be noted that none of these tables include priorities, but these are implicitly given via the requirement specification (table 5.1). The only requirements not included in the test plan are those with low priority, as they are considered outside the scope of the proposed system solution and thus not interesting from a test plan point of view. The last test (T12) relates to requirements with different priorities, but it definitely has high priority. (Without the system functioning as a whole, it is, in all fairness, rather useless.) The test plan will be revisited in chapter 6, where testing of the system parts which have been implemented will be presented in more detail. The two GIS solutions, which relates to only a few of the tests presented here, will probably need their own internal test plans based on their complete design. Since no complete design for these parts have been presented here, such test plans are also seen as outside the scope of this thesis.

[3]I/S is short for Implemented/Successful. The answers will be Y (yes), N (no) and P (partially)

Test nr	**Req nr**	**Description**
T1	R1, R2, R5, R6, R7, R10, R11, R12	Data updates via central application GUIs result in correct database changes.
T2	R3	Job status update using field application GUI results in correct database changes.
T3	R4	Possible to retrieve information about a given job at any time.
T4	R8	When a driver logs into a field application, the driver and truck is associated to one another in the database.
T5	R9	The field application works if, and only if, there is exactly one driver logged in.
T6	R13	When optimisation process is initialised in GUI, it is executed correctly.
T7	R13	Optimisation is carried out properly.
T8	R14	The position of a truck according to the GIS is correct.
T9	R15	The field application GIS offers correct directions from current position to a fixed point.
T10	R16	When a truck is assigned to a route, the route information shown in the truck must be correct.
T11	R17	Data communication works properly.
T12	All	Complete system test.

Table 5.3: Test plan – description

Test nr	How to perform test	I/S
T1	Use GUI to insert data, and control their correctness in the database.	
T2	Insert data via GUI, control correctness in database.	
T3	Retrieving job information should be available via the central application GIS, correctness of data must be tested by validating database queries.	
T4	Login via login window in the field application, control correctness in database.	
T5	Ensure that it is only possible to do field application tasks when a driver is logged in, and that no driver is allowed to log in if another driver is already logged in.	
T6	Check, using i.e. trace outputs from process, that the optimisation process is started correctly.	
T7	Test heuristic correctness by using a small, known dataset with stepwise output.[4]. A large random data set should also be tested, to ensure acceptable runtime.	
T8	Check if position in GIS corresponds to actual position by placing field application at a known address.	
T9	Test driving instructions in real life between two points where the shortest route is known.	
T10	Create a known route, send it to the field agent application, and check that it appears correctly.	
T11	Check that connections are set up in the correct way, and that the messages sent are treated correctly.	
T12	When all tests T1–T10 succeed individually, the complete system should be tested in practice for desired functionality. While the other tests ensure that key functionality works as expected, this integration test is of course needed.	

Table 5.4: Test plan – execution

Chapter 6

Implementation

In this chapter the system implementation will be discussed. This means that what has been implemented will be presented with regards to how it works, and what programs have been used to develop the different application parts. The author would like to point out that presenting the programs used are done for the purpose of adding depth to the implementation process description only, and should not be seen as a software recommendation. Whenever such software is presented, the reason behind the selection will be pointed out. In addition to dsecribing the implementation already done, this chapter will also present what must be done to complete the system and how these implementations could be taken care of. The test plan from chapter 5 will be revisited, with details and results from the performed tests.

6.1 Implementation Presentation

This section will present the existing implentation. Before, and during, the implementation process taking several important decisions about the development environment has been necessary, and these will be presented first.

Programming language

The first decision to take in the implementation process is to choose a programming language for the system development. There are several languages which could work nicely, but Java was chosen for the following reasons:

- Platform independent. It is desireable that the system may work on any platform, under no other restriction than that it supports Java. This support is available in all common operating systems on fixed platforms, and on an increasing amount of operating systems for mobile

units. This independency keeps the system from being locked to specific vendors, which was one of the charachteristics the system was meant to have. (The independency of the solution will be further discussed in chapter 8.)

- Author familiarity. Java is the programming language the author is most familiar with, and due to the time restrictions on this work this is highly preferable from an efficiency point of view.

Java is developed by Sun Microsystems, and documentation of all sorts may be found at their Java website ([29]).

Java consists of three platforms: Enterprise (J2EE), Standard (J2SE), and Micro (J2ME). For the central application J2SE has been used. It offers all the standard features necessary for the desktop (or server) solution, such as advanced GUIs and connectivity. It is possible that using J2EE functionality could have simplified parts of the solution implementation (such as i.e. database connection), but the functionality of J2SE has been found sufficient for this implementation process. In the field application, which it has been decided should operate on a small mobile device, in most cases a PDA, the J2ME platform has been used. J2ME is made up of a set of APIs (Application Programming Interfaces), and the main API in the field application is called CDC (Connected Device Configuration). This is the more advanced of the two main APIs in J2ME (the other one is called CLDC (Connected Limited Device Configuration), and it is meant for more advanced mobile entities with a certain level of processing power, such as PDAs and tablet-PCs. Using CDC should provide the technology necessary for implementing the field application, with regards to connectivity and user interface.

Development environment

To simplify the process of creating the system, it has been programmed within an Integrated Development Environment (IDE). The IDE chosen was the NetBeans IDE, which can be found at [30]. There are two main reasons behind choosing this environment, and these are:

- Simple but powerful GUI development. NetBeans offers a drag and drop based GUI builder, where you simply choose components, place them as you like, and update names and properties via schemes and dialog boxes. The code is generated automatically based on the user's preferred design. Such functionality is very helpful since GUI building by coding is a tedious task, consisting of a lot of codelines.

- Free. The NetBeans IDE is freeware. Having such an advanced tool at your disposal without costs or limitations, is a big plus. While other very good IDEs exist, they usually require a solid investment or settling for a free/demo edition with limited functionality.

An example of what the NetBeans IDE looks like while used can be found in figure 6.1.

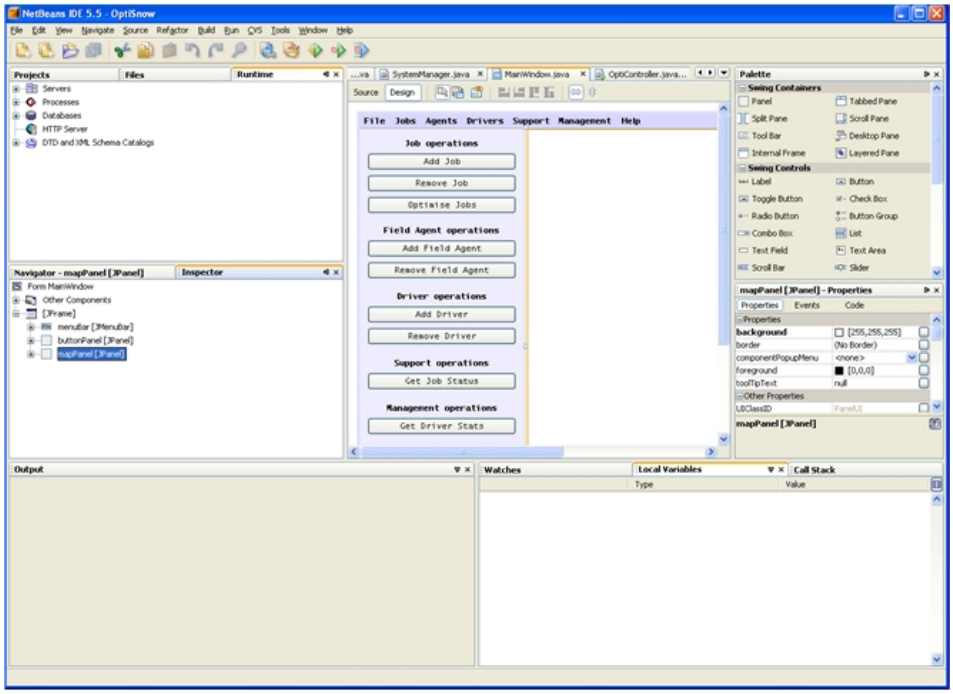

Figure 6.1: NetBeans IDE screenshot.

Database

To create the database for the proposed system, the Derby database from Apache has been used. This is a relational database implemented entirely in Java, and was chosen because it was easy to use with JDBC (Java Database Connectivity). (The Derby database is actually used by Sun in their own Java Application Server.) For more on the Derby database, see [31].

6.1.1 GUI implementation

In this section both GUI implementations will be presented. Since the GIS solution is not implemented in neither of the applications, what will be offered here is the proposed interfaces in the system.

Central application

The focus point for this GUI was offering an easy to use interface towards the operators. All the basic functions, such as adding and removing entities, are available as both via buttons in the window and via the menu. (While this is the solution in the current implementation, maybe a solution with no button and only the GIS in the window could be a better solution.) In addition, the menus contain the options necessary for functions typically included in a complete solution, such as help functions, which have not been implemented here. The implemented main window can be seen in figure 6.2. (The large white area is meant for the GIS.) The GUIs for adding and removing the different entities has also been implemented, and an example, the add job window, can be seen in figure 6.3. The other interaction GUIs look similar to

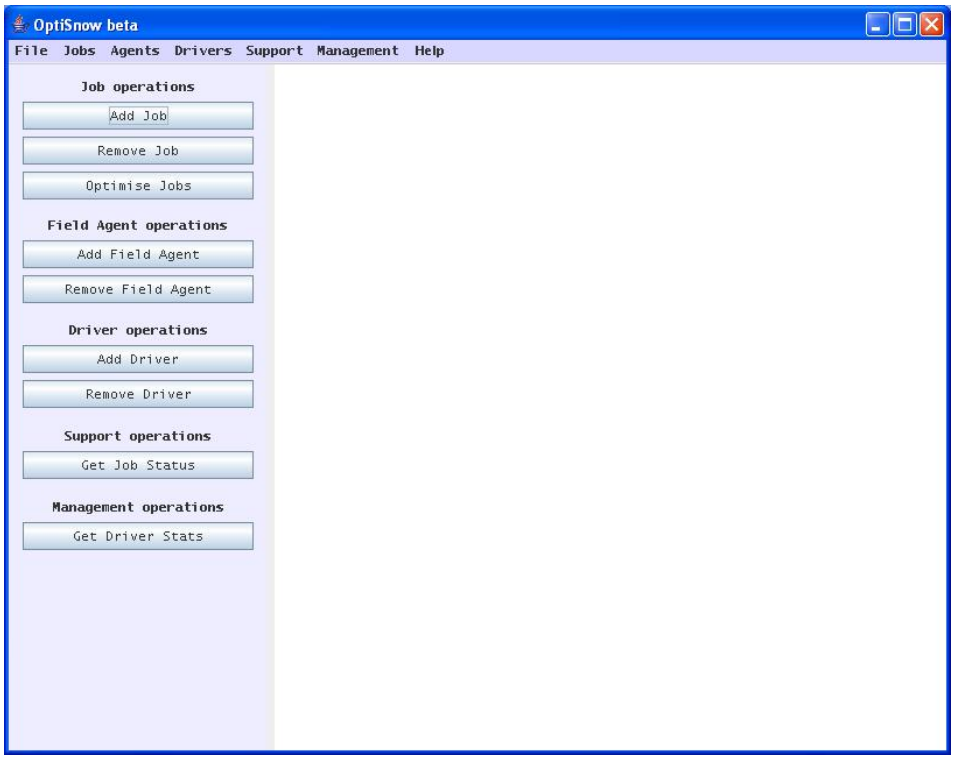

Figure 6.2: MainWindow GUI

the add job window, and should be very easy to understand for the operators. The main window is implemented using the JFrame class in java as the top level container, while the dialogs are realised using the JDialog class on top. It should be noted that the ”Get Job Status” button included in the current version of the main window will probably be obsolete, as this functionality is

Figure 6.3: AddJobWindow GUI

likely to end up within the GIS.

Field application

The field application GUI is much simpler; with the main window having three alternative views, the GIS, the list of jobs and the login window. Below all views information about current driver and job is listed, and the button for changing job status accesible. The three views can be seen in figure 6.4. (An alternative, possibly better, solution would be to have the login window show up on start-up, but the author had troubles getting this to work using J2ME.)

Figure 6.4: Field application GUI. Map view (left), joblist view (centre) and logon view (right).

6.1.2 Database implementation

The database has been implemented exactly as it is depicted in figure 5.3. All fields which cannot be empty have the NOT NULL constraint, and all fields which must be unique, such as the keys and the truck registration number, have been marked with the UNIQUE tag. All tables of course has a primary key, and the necessary foreign keys. It is worth mentioning that Derby does not support the AUTO_INCREMENT option that some other databases (like i.e. MySQL) have. This option is very practical for automatic key value generation, and to obtain this effect in this database the keys were created using the following SQL () statement:

```
id INT generated by default as IDENTITY(start with 1)
```

When this code is included in the CREATE TABLE statement, the id field will be created as an integer, the first entry will get value 1, and the value is then incremented by 1 for each new entry. Apart from this there really is nothing special about the creation of the database, just simple SQL statements are needed. For at detailed guide on how to use Derby, see [32].

For the database to work it must have an IP address. In the system implementation the IP is simply set to the host 'localhost', which always refers to the current connection the computer it runs on is assigned to. In a commercial system it should be placed at a separate place, and its IP should be hardcoded into the central application. There are several reasons why this should be done. First of all, it ensures that the database is not affected by prospective errors in the central system. It also means that the database can be placed in a physically safe place, lowering the risk for someone to unintentionally alter it in any way. In short, a solution like this is the best way to ensure that the database functionality is maintained, which is very important for the system to operate as required.

6.1.3 The core functionality

In the core of the two applications, the functionality is implemented as presented in the design phase. The presentations are generally short, but the optimisation implementation will be discussed thoroughly.

Central application

The central application consists of five classes. This is what they do at current implementation status:

- DatabaseHandler. Sets up an ODBC connection with the database, gets, updates, adds and removes rows from the database tables based on received queries.
- CommInitiator. Sends messages to field agents on request.
- CommListener. Listens for incoming messages on a given port.
- OptiController. Carries out optimisation process, more info below.
- SystemManager. Uses DatabaseHandler to update database based for all update requests received. Initiates optimisation process in OptiController when OptimiseJobs button in GUI is pushed. It should also see to that all routes are then correctly distributed, but this is not currently implemented. In short, this class represents the intelligence in the system.

When it comes to the optimisation, it has been treated as the main point for this system implementation. Therefore, two heuristics have been implemented to see whether one is superior to the other, or if the result depends on the dataset. The heuristics chosen are listed below, along with an explanation of why the were picked:

- Farthest Insertion. A modified version of the TSP heuristic farthest insertion to allow for several travellers has been implemented. It has been chosen because it gives acceptable results, and because it is well known to the author. The last reason is especially relevant when testing that the heuristic implementation runs correctly. Alternatives for the TSP approach would have been other insertion algorithms, but they are, statistically, worse, or Christofides', which is significally more complicated to implement.
- Clarke & Wright. The Clark & Wright algorithm is widely used in fleet management even today, despite being more than 40 years old. The reason is that its performance versus runtime and complexity output is still highly attractive, and also the reason why it has been chosen in this case. Alternatives among the VRP-focused algorithms would have been the Fisher & Jaikumar algorithm, but it is much slower and more complex, and the sweep algorithm. The latter's use of angle restrictions seems rather unfitting for large problems (results in very small angles).

How these heuristics will be tested, the test results, and a recommendation for which solution to stick with, will be described in the section about testing

later in this chapter. It should be noted that modern approaches, i.e. search methods, provide more accurate results. In large applications planning for longer time periods, implementing such a solution may be the best option. In such a case the time used for optimisation (which is substantially longer for these new methods) is less important. But for a scenario such as in our case, where optimisation is stated as a daily operation (maybe even several times a day) speed is of the essence, and the lesser accuracy acceptable.

The reason for doing this testing is to do research for the complete solution. Since no GIS will be implemented, completing the optimisation part of the system completely will not be possible. It then makes more sense to focus on the heuristic quality in the optimisation process, since it is one of the core functionalities the system offers.

The functionality for treating optimisation of jobs added after the first optimisation run of the day, has not been dealt with in the current implementation. This problem will be thoroughly discussed in the discussion in chapter 8.

Field application

The core in this application contains the same connection functionality as its central counterpart, and it also takes care of driver login. To ensure system correctness, the buttons in the window have been instructed to only perform their tasks if one driver is logged in (except the login task), and to disallow any login attempts if a driver is already logged in. The joblist management functions have not been implemented, since the required functionality in the central application is not in place.

6.2 Further Implementation

The implementation done here simply fullfills some of the requirements for the system, and a lot of additional work remains to complete a fully functional system. This section will present some of the work that needs to be done, and how it can be implemented.

6.2.1 GIS implementations

The main vital part remaining of system implementation concerns the two GIS solutions. Especially the central application GIS is critical for the system. Since no complete design for the GIS exists, there is not much more to

say about what functionality must be included. What should be noted is that the GISes must be compatible for use in a Java application. An example of a possible environment for developing the GIS solutions is called ArcGIS and delivered by an American firm called ESRI (Environmental System Research Institute). This program already contains a large database and offer developers the possibility to make both their own separate GIS solutions using the existing database, or simply creating extensions to the already existing functions in the program. Developers may use a wide variety of programming languages to develop their solution, among them Java. (For more on ArcGis, see [33].) ArcGIS is not the only option fo such an environment, and is only used as an example. If a GIS is to be implemented, it is of course necessary to consider the choice of environment with much greater depth.

6.2.2 System integration

When the two GIS solutions have been implemented the last, and in many ways most important, phase of implementation takes place; the integration. While some functionality can be completely testet with out the GISes, most parts depend on it. Putting all parts successfully together, removing errors of all sizes, can be tedious work, but if it is done carefully, the system should be well functioning. After this final part of implementation, the system should be thoroughly tested as a whole, both with simulation and in a realistic real-world setting.

6.3 Testing

Here test results with respect to the requirement specification will be presented. First however, the testing for an appropriate algorithm for the optimisation process will be introduced.

6.3.1 Choice of optimisation algorithm

Test layout

As mentioned in the previous section two algorithms have been tested, Farthest Insertion with VRP adaption, and Clarke & Wright. The testing was done in the following manner:

1. One fixed, small test. A 15-node, 3-truck example using a fixed cost matrix and uniform time demands for jobs where used for the first test. This test was used mainly to observe the correctness of the algorithms.

By inspecting the stepwise process in choosing and inserting the next node, and comparing with the expected behaviour, observing this, and thus validating T7, was possible. The result was interesting as well, since it was the only test where both algorithms offers a valid solution. The test set had its nodes placed in what could be seen as a three-leaved clover pattern, with 5 nodes on each leaf.

2. Ten semi-random tests. 300 jobs and 25 trucks were used for this test, where distances where randomised but demands still fixed and uniform. This test allowed comparison on a dataset of more realistic size, but without breaching the total capacity available (supply = demand).

3. Ten random tests. Similar to the previous, but demands are here randomised as well, making it possible that demand is larger than supply. In the two latter cases several runs were executed to look for performance patterns.

The results from this testing can be seen in table 6.1. Notice that it is noted in the table if each algorithm made a successful run as well, which will be discussed later on.

Test results

The main thing the table shows is that Clarke & Wright outperform Farthest Insertion, on average (based on all tests) the result of the latter is 12.5% worse. There are however a few things more things that must be explained, especially concerning the errors.

First of all, the reason for the 'Yes' in the error column for FI is based on the fact that it does not have capacity constraints. This is not a standard feature in this algorithm due to its TSP nature. To implement FI with capacities would require that for each time a node outside the tour is checked against a node inside the tour, it must also check if the route capacity, meaning the capacity used between to depot nodes in the tour, which adds a whole lot of extra calculation (since the position of all depot nodes in the tour must be known at all times). But the capacity constraint should not lead to a better FI solution value. The reason is simple: In the test cases there are often around 20 of the 25 depot nodes in a row, meaning that in the FI solution that many zero distances will occur. (The solution distance is estimated when the extended route array has been transformed back to the standard problem, thus the infinity lengths between the (virtual) depots are reomoved.) As a result of this the acquired solution in this case will have

Test	Farthest Insertion	Error	Clarke & Wright	Error	Ratio
1.1	81	No	72 (optimum!)	No	1.125
2.1	1987	Yes	1579	No	1.258
2.2	1877	Yes	1669	No	1.125
2.3	1876	Yes	1666	No	1.126
2.4	1881	Yes	1668	No	1.128
2.5	1761	Yes	1620	No	1.087
2.6	1770	Yes	1567	No	1.130
2.7	1764	Yes	1629	No	1.083
2.8	1624	Yes	1560	No	1.041
2.9	1920	Yes	1682	No	1.141
2.10	1712	Yes	1589	No	1.077
3.1	1987	Yes	1687	Yes	1.178
3.2	2006	Yes	1706	Yes	1.176
3.3	2057	Yes	1709	Yes	1.204
3.4	1916	Yes	1560	Yes	1.228
3.5	1813	Yes	1622	Yes	1.118
3.6	1718	Yes	1620	Yes	1.060
3.7	1758	Yes	1671	Yes	1.052
3.8	1864	Yes	1483	Yes	1.257
3.9	1930	Yes	1764	Yes	1.094
3.10	1599	Yes	1426	Yes	1.121

Table 6.1: Result comparison. Farthest Insertion and Clarke & Wright.

an extra length of around 160 (the estimated value of the random distances used are 8 in this case (randomised between 1 and 15)), in addition to the potential. In addition to this comes the possible worse solutions necessary due to the restriction on allowable inclusions, but the result could just as well be better (since the new insertion pattern could end up being better at completion).

It should be noted that these shortcomings are not algorithm errors as such. The implementation works completely correctly according to the algorithm statement, and has even been enhanced to work well with the virtual depots. This has been done by checking all inclusion possibilities next to all depot nodes every time a depot node has been found as the farthest away nearest node in the tour. The problem occurs when the depot is not the point of insertion at least as many times as there are depot nodes, which keeps all links between depot nodes from being removed.

The 'error' column for Clarke & Wright does not refer to an actual algorithm error either. It simply says that in this case it was not able to complete all jobs. In the 3.x tests the demand is statistically a bit higher than the supply. This leaves some jobs which cannot be executed. The algorithm itself works correctly (verified by inspection after test 1.1), but the system should be made aware that there are unassigned jobs. In a complete version of the system there should be some functionality taking care of this problem. It would for example be unfair if a job which was registered early was not done but a job registered late was, as a result of the optimisation process. This could be taken care of by i.e. letting the system remove the most recently added jobs from the current optimisation process and then running it again. In this case the system would of course have to notify the operators of its decision, so either the customers can be informed that they won't get snow clearing or the operators open up to using overtime. In the latter case the optimisation process should be run again with extended truck capacities.

Please note that the CW implementation differs from the algorithm scheme presented in chapter 3 in one way: In the algorithm statement it says that if there is no positive saving left, the algorithm should terminate. This would lead to jobs whic potentially may be executed, not being executed due to the fact that all remaining jobs are far apart. It is a problem in CW, at least in the sequential way it has been implemented here, that the final tours may be a lot less effective than the first. In this case the restriction to exit when savings drop below zero has been dropped, and the only exit trigger in the implementation takes place when all trucks are used to their limit. To avoid such bad late routes, a special scheme may exist when the savings drop below zero, but the author has not been able to find, or come up with, such a scheme himself.

Choice of algorithm

In the end the system should only contain one algorithm, and though the testing, due to the FI implementation's lacking of capacity constraints, certainly has its weaknesses, the Clarke & Wright approach is recommended. In addition to performing better on all tests (though the testing basis statistically is small) CW has better result expectancy with regards to optimum for the VRP than FI has for the TSP (see chapter 3), and there is no suggesting that a TSP approach gives better results for the VRP than for its original problem. Clarke & Wright gives goods results with regards to the global optimum, and it is also preferable with an algorithm that does not needs

much alteration in practical use. Both algorithms perform the 300-node, 25-traveller test problem in virtually no time, so performance on large datasets are acceptable for both and does not favour one over the other. Bottom line: CW is a solid choice.

As an end note about optimisation the author would like to mention that the TSP approach could have given other results by altering the dataset. By adding the demand of each job to the cost of reaching it from all other jobs, an assymetric cost matrix can be created. This would require a slight alteration of the code, since c_{ij} is no longer the same as c_{ji}, but would include demands into the tour creation process. While it would not change the fact that a time constraint for each subtour must be enforced, it would be interesting to see what positive (or negative) effects it has on tour creation. (Due to time constraints the author has not been able to implement this solution.) For both algorithms using a post optimisation method should also be considered, as a method such as 2-opt could significantly improve results.

6.3.2 General test results

In the current state of implementation, the following tests have been possible to perform to some extent: T1, T4, T5, T6, T7 and T11. All add and remove options in the field application works properly, but since the GIS has not been implemented this does not include the distance array for each job. In addition, possible updating of driver and truck characteristics, mainly driver address, has not been implemented. T6 concerns initating the optimisation process via the GUI. This works within the current boundaries of the system, but must be adapted when real job data is to be used. (Currently it only initates test data.) Both these tests have been verified by running the central application. For facts about optimisation algorithms see the previous section. In short, they work as expected, but adding further functionality may enhance the process further. T4 has been tested by trying to make a driver login both with correct and incorrect data, but database status updates are not implemented so only the password check has been tested. T11 has been verified based on the correctness of the T4 test (though at the moment communcication is not done via secure sockets). T5 has been fixed by simply using a boolean value in the field application main window, set to true when a driver is connected and false otherwise. Then all interaction possibilites in the window has been executed based on the value of this variable.

For the test not performed, the reasons are as follows:

- T2. Job lists not yet implemented.
- T4, T8, T9 and T10. GISes not implemented.
- T12. System not complete.

Based on this, let us revisit the test plan presented in chapter 5, found in table 6.2.

Test nr	**How to perform test**	**I/S**
T1	Use GUI to insert data, and control their correctness in the database.	Y/Y*
T2	Insert data via GUI, control correctness in database.	N/-
T3	Retrieving job information should be available via the central application GIS, correctness of data must be tested by validating database queries.	N/-
T4	Login via login window in the field application, control correctness in database.	Y/P
T5	Ensure that it is only possible to do field application tasks when a driver is logged in, and that no driver is allowed to log in if another driver is already logged in.	Y/Y
T6	Check, using i.e. trace outputs from process, that the optimisation process is started correctly.	Y/Y*
T7	Test heuristic correctness by using a small, known dataset with stepwise output.[1]. A large random data set should also be tested, to ensure acceptable runtime.	Y/Y*
T8	Check if position in GIS corresponds to actual position by placing field application at a known address.	N/-
T9	Test driving instructions in real life between two points where the shortest route is known.	N/-
T10	Create a known route, send it to the field agent application, and check that it appears correctly.	N/-
T11	Check that connections are set up in the correct way, and that the messages sent are treated correctly.	Y/P
T12	When all tests T1–T10 succeed individually, the complete system should be tested in practice for desired functionality. While the other tests ensure that key functionality works as expected, this integration test is of course needed.	N/-

Table 6.2: Test plan – results. * - alteration may be needed in final version.

Chapter 7

Business Model

This chapter will present a business model for the proposed solution, or rather for a company centered around developing and selling the proposed solution.The model will be developed based on an ontology proposed by Osterwalder in [34]. The first section will present the different parts of the model for the company, while the second section binds them together and shows the complete model. The last section of this chapter offers a short discussion on how implementing a software such as this may affect the business model of a snow clearing company.

7.1 Model Building Blocks

Osterwalder determines nine main areas that need to be included in a business model, and these are:

1. Value proposition. Presents the company's products and services.
2. Target customer. Which customer segments does the company offer its assets to.
3. Distribution channel. How the company plans to contact customers.
4. Relationship. What links the company has to its customers.
5. Value configuration. The company's arrangement of activities and resources.
6. Core capability. Which competencies the company needs to do business as desired.
7. Partnership network. Describes the partner network used to offer value.

8. Cost structure. Describes costs the company will have.

9. Revenue model. Describes how the company makes money.

These nine points are divided into four groups: Product (1), Customer Interface (2, 3 and 4), Infrastructure Management (5, 6 and 7) and Financial Aspects (8 and 9). All points will now be evaluated, and the complete model showing their relationships will then be presented. All attributes presented in {} are taken from Osterwalder, and further description can be found in his thesis ([34]). Please note that all tables in this section, with the exception of table 7.2, are also based on tables in Osterwalder's thesis. (The level of table modification varies, some are equal but with different data, while others are more significantly altered.)

7.1.1 Product

Value proposition

The following products and services, called Offerings in the ontology, can be delievered by the company: The system, system updates and support. Both system and support mainly offer value when used, so they belong in the category {Value use}. While updates secures value as a result of renewal, and belong in the category {Value renewal}. None of these products can be called innovative or high-end, so they belong in the category {me-too}(Me-too implies a rather standard product). Since the system is relatively small, and has limited functionality compared to many other fleet management solutions (see chapter 8, let us price it at the {economy} level, below the standard market price, and do the same with support. To make the system attractive the updating can be offered for free, and we end up with the value proposition seen in table 7.1.

7.1.2 Customer Interface

Target customer

The target customer for the system is described in table 7.2:

Distribution channel

Let us assume that the company has an office and a website, and that these are the main contact points towards the public. In addition both the web

	System	**Updates**	**Support**
Description	The system offers the buying company a fleet management system offering route optimisation and fleet control.	The system is regularly updated to remove errors and add new functionality.	Support functionality like training and troubleshooting.
Reasoning	Better fleet management may lead to a substantial cost savings.	Updates enhances the system, thus adding value.	For maximum system utilisation support functionality is required.
Value life cycle	{Value use}	{Value renewal}	{Value use}
Value level	{Me-too}	{Me-too}	{Me-too}
Price level	{Economy}	{Free}	{Economy}

Table 7.1: Value proposition.

Name	Snow clearing company
Description	The company sells a product targeted at snow clearing companies that want to increase their operational efficiency.

Table 7.2: Target customer.

and newspapers are considered helpful in the {Awareness} phase, when potential customers should notice the product. Table describes the distribution channels.

Relationship

Relationships are connected with three customer equities: acquisition, retention, and add-on selling. The two first are interesting in this case. The company needs a strategy to obtain new customers, and one to retain those it already has. In the aquistion phase, possible solutions may be i.e. a free test period. For retention the most important part is probably solid support offerings, does obtaining customer trust. Table 7.4 sums up these relationship ideas.

Channel	Awareness	Evaluation	Purchase	After sales
Homepage	Product advertising.	Company information, product catalogue.		
Office	Product advertising, promotion.	Product catalogue, specification, test application, advice.	All parts of the sales process.	Training and support.
Newspapers	Advertising			
Websites	Advertising			
Customer sites				Training and support

Table 7.3: Distribution channels.

Customer Equity	{Acquisition}	{Retention}
Relationship description	Try to acquire new customers by allowing them to test it themselves in their own environment.	Try to keep customers by offering state-of-the-art support.
Relationship name	Free trial	Quality support
Relationship mechanism	Customers get one month free trial before they have to pay for the system.	Support is quickly available at any time.
Reasoning	Use: Customers get to evaluate the quality of the program in realistic settings.	Risk: Problems can be dealt with quickly at any time of the day, possibly reducing operational losses.
Function	-	Trust

Table 7.4: Relationships.

7.1.3 Infrastructure Management

Value configuration

The company can be modelled as a value chain. It produces a product and sells this to its customers, similar to how traditional manufacturing compa-

nies work. While it offers a network service, this is only an internal system tool and thus the value network profile does not fit. To model the value network, Osterwalder creates activities, and relates these to one of five phases (in the case of a value chain): Inbound logistics, operations, outbound logistics, services and marketing. Table 7.5 shows how this can be modeled for our company. (This is done on a high level, and several of these actions could be split up into seperate ones.)

Activity name	**Activity description**	**Level**	**Nature**
Acquire resources	Invest in any software needed, plus all equipment needed for producing end product package.	Primary activity	Inbound logistics
Engineer solution	Program the system, create installation disc.	Primary activity	Operations
Pack and deliver system	Ready and deliver end product package.	Primary activity	Outbound logistics
Provide support	Offer the described support services	Primary activity	Support
Advertise	Do advertising as described	Primary activity	Marketing

Table 7.5: Value configuration.

Core capability

The core capability of the company describes what resources it has available. This resources can be goods (called tangibles), personnel, and other resources such as a strong brand name (called intangibles). For our company, these core resources are summed up as in table 7.6.

Partnership network

A partnership network is a set of agreements made between companies to set up deals which are seen as mutually beneficial. For our company no such partnership is a must have. The support function may be looked upon as a partnership, but it is rather one-sided. There are other possibilities though: If a large customer is acquired, like i.e. a company responsible for all public

Name	Description	Type
Hardware	All hardware necessary for creating the system, and making it ready for shipment.	Tangible
Software	All software necessary for creating the system, and making it ready for shipment.	Tangible
The system	The end product ready for sale.	Tangible
Programmers	The personnel needed to develop and later update the solution.	Human
Consultants	The personnel required to carry out the support functions.	Human

Table 7.6: Core resources.

snow clearing in a large community, it could be interesting to create a testing agreement. The customer could for example get the product at reduced cost, if they entered into an evaluation agreement where they report back on test results. Such an agreement could benefit both companies. While the customer could reduce its software costs, it could also get a better program through this active participation. A better software will benefit our company as well, as it may attract new customers.

7.1.4 Financial Aspects

Cost structure

The cost structure consists of a number of accounts related to one source of expenses related to the company. Each of these accounts are related to a monetary sum, and a percentage showing how much of the total cost it makes up. For our system, exact sums are hard to offer, and only a percentage suggestion is included. The costs can be seen in table 7.7.

Revenue model

The company will have income from two of its operations: Selling the product and providing support. (Remember that updates were set to be free of charge and will not be seen as revenue adding, even though it may lead to increased support income.) The product could be sold at a fixed price or at a price dependent on the size of the consumer (number of field vehicles). In this model, the fixed price alternative is chosen. (This should be studied with more depth in a real-world implementation.) The support function will

Account	Description	Monetary size	Share
Purchases	Buying all needed equipment	-	15%
Development	Developing and updating the system	-	30%
Support	Support related costs such as i.e. travel costs	-	15%
Fixed	All sorts of administrative costs.	-	20%
Advertising	Cost for all advertising	-	15%
Other	Unforeseen costs	-	5%

Table 7.7: Cost structure.

typically be differentiated based on how often support is required. Having a fixed basic price i.e. per year for offering the support service is definitely a possibility,though. How much each part makes up of the total revenue is hard to say, but lets estimate that selling the product make up 60%, and support the rest. Both operations are based on selling; one of them the product, the other human resources. Table 7.8 sums up the revenue model.

Source	Description	Stream type	Pricing method	Share
Sale	Income from selling the product	Selling	Fixed	60%
Support	Income from offering product support	Selling	Differentiated	40%

Table 7.8: Revenue model.

7.2 The Complete Model

When combining all presented parts into a complete model, we get an overview of how the company operates. This information should be very useful when setting up and running the business. The complete model for our system can be seen in figure 7.1.

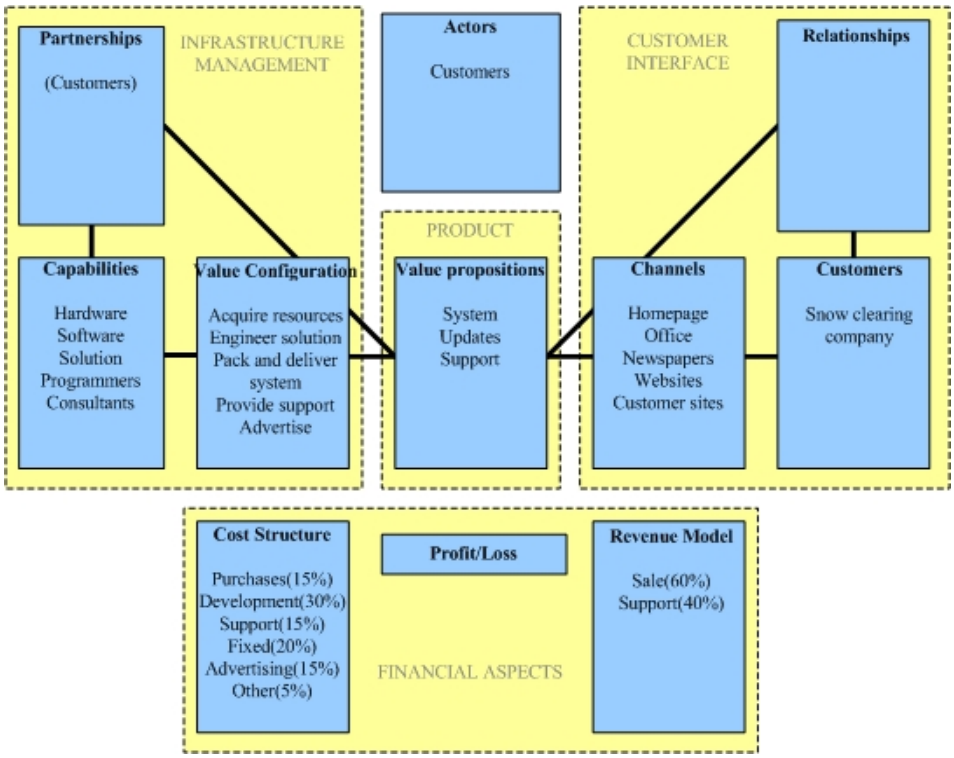

Figure 7.1: The Complete Business Model. (Based on a figure in [34].)

7.3 The Buyer's Perspective

As well as constructing a business model for the company selling the proposed solution, it is interesting to see how it affects the customer. No complete business model will be given for the buyer side, only a short discussion of some points.

Competency

Using the proposed system a snow clearing company may add or update a few things to their core competency summary. As mentioned earlier in this text (see i.e. chapter 5), the increased transparency a fully developed version of the proposed solution offers can lead to a better customer interface. Offering customers detailed support on request is definitely a competency worth mentioning. The company can now state not only that it offers snow clearing, but also very effective snow clearing. Knowing one's competencies is important when developing a market strategy, and implementing the proposed solution into its business could give a company offering snow clearing

a few new strings to play on.

Costs and revenues

While investing in the system certainly brings new costs, both as an initial investment and for continuous suport, other costs should decrease. By being able to do work more efficiently, the operational costs, like i.e. gasoline spendings, could decrease. Improved communication can also reduce costs in case of operational complications by reducing response times. For companies offering public snow clearing (often this may be public offices themselves) jobs *have* to be done. This means that on days with heavy snow falls, overtime is required. Effective work will lead to quicker job execution, which again suggests lower personnel costs. While there are no guaranties, there is certainly a possibility that the cost structure will not only change but its total monetary estimate will decrease.

A company offering snow clearing makes money by executing jobs. The more efficient it is the more jobs it is capable of accepting. It should then be clear that a more efficient job scheduling process, using the proposed solution, would lead to higher income, dependent of course of market demand. (If the company already covers all accessible demand, revenues may not increase.) Better terms of service may also implicitly increase revenues, by attracting new customers.

Whatever situation the snow clearing company is in it should, if after thorough analyses it expects the revenue-cost balance to increase by implementing the proposed solution, go forward with the system acquisition.

Chapter 8

Discussion

In this chapter the proposed solution will be thoroughly discussed. A quality assessment will be made, along with a view at what may complicate the solution and require extensions. The solution will also be studied from a market point of view. This includes assessing which other businesses could be interested in the solution and comparing the solution with existing fleet management solutions. In addition to this, future work needed for adjusting the solution to its real-life purpose will be presented.

8.1 Solution Quality

First, let us assess what the proposed system can offer a company. When fully implemented, the following features should be included:

- Heuristic based route optimisation.
- Detailed, real-time information on the company's operation.
- Increased operational transparency.
- Driver support.

The optimisation is done using an heuristic approach. Such an approach is time efficient even for large problems, but does not offer a complete solution. The solution offered wil however normally be less than ten percent larger than optimum, which is substantially better than what can be expected with a non-optimisation based route creation scheme. Since the number of customers to be served could be very large, the savings may very well be substantial. The solution is also obtained very quickly, and thus it is user friendly.

The graphical interface offered to operators should simplify their work significantly. Having easy access to every driver's position and job's status could be of great help both when supporting customers and in case of an unexpected event. How well these functions work does depend on the users' understanding of how the system works though, and to be successful solid training is required. If the users are well trained, and understand and appreciate the system, then it should be a valuable asset to the company.

A complete system based on the proposed solution could support advanced management functions based on i.e. performance statistics. With such statistics present, the management has a useful tool in assessing the company's general and an employee's personal performance. Such information can be used to i.e. discover weak points in the company's operation, which then may be assessed.

The drivers benefit from the solution by receiving complete information about their work days. The job list received in the morning could contain estimated time per job, allowing driver's to plan i.e. breaks at the start of the day. The driving assistance could also be of help, especially when a driver covers new areas. There are many functions that are included in existing solutions in the market that the proposed solution does not support in its current state. Some of these are:

- Management and customer functions. Some of the functions which have been mentioned during the course of this text concerning management and customer functionality should be included in a complete solution.

- Truck lifespan functions. Commercial fleet management systems often include not only operational fleet management, but also fleet management with respect to i.e. truck maintainance and other equipment control functions. The proposed solution is generally functioned on one task (operational fleet management) and thus has a rather restricted area of use.

To conclude this section, it can be stated that: The proposed solution could enhance operations for the snow clearing company, but it could also be seriously enhanced itself.

8.2 Solution Challenges

There are many abilities that could be demanded from the solution in addition to those included within the limits set for the proposal presented. This especially relates to the optimisation process. Below, some of the additional demands that can be put on jobs and trucks are presented along with the complications they bring along.

- Job priorities. There may be priorities among the jobs. If one of the jobs is important to secure public safety for example, it is likely to have some sort of priority. Such priorities restrict the number of possible routes, and add constraints to the problem. This means that the optimisation process must be expanded to take these priorities into consideration.

- Job time windows. Some jobs may need to be taken care of between two given points in time. Such time windows complicate the optimisation in much the same way as priorities, and must be taken into consideration when implementing the optimisation process.

- Special demands. Some jobs may require special features, such as i.e. road sanding. These resources may be available in limited amounts per truck, and will add an extra node capacity constraint for the routes being created.

- Heterogenous fleet. In the case of special abilities such as the road sanding ability mentioned in the previous point, trucks may have different capabilities. Such differences must also be taken into consideration in the optimisation model, adding further constraints.

- Distance between two nodes dependending on direction. It is not unlike that traffic from point A to point B requires either more or less time than travelling in the oposite direction. This could be the result of one-way streets, rush hour, etc. This could affect the optimisation process, as the cost matrix changes.

- Multiple depots. A large company does not necessarily have only one garage for its trucks, leading to multiple existing depots. If this is the case, the optimisation process will require reengineering.

- Other optimisation goals. The goal for the optimisation process may not be using the least time, but maybe travelling as short as possible, using the least gasoline, etc. This type of changes should not lead to

a very different solution process, it simply requires different data sets as a basis. If the goal instead becomes using the least trucks, then the formulation will change some, but not much if using the currently implemented version of Clarke & Wright.

This list shows the diversity and difficulty related to advanced vehicle routing. While some of these are likely to be essential in a commercial solution, they have not been included in the proposed system. The reason is that all they really do is complicate the implementation process, without really changing the basis of operation. For inspecting the problem the proposed solution should be advanced enough, but the points in this list must not be forgot in a solution up for release. There exist a lot of literature looking into these extra constraints, for an example concerning time windows, see [5].

Optimisation during the day

It has been noted several places in this text that some special routine is needed for optimising new jobs added during the day. There are several reason why, and some of these are:

- Trucks are not at the depot. The optimisation process is based on a the presumption that all trucks are currently at the depot, which is only the case at the beginning of the day. Without the presumption in place the optimisation process will be less accurate.
- Drivers want consistency. Reconstructing driver routes during the day is not likely to be popular in a real-world environment. When a driver gets a job list in the beginning of the day, you cannot expect him to accept extensive changes during the day, as this could be seen as limiting his control over his own work.

While no concrete solution for this problem has been proposed, the author would think that a scheme were jobs are added to the end of current job lists based on these lists current status would be the best approach. This way jobs can both be assigned to trucks in the right area, and without invoking comprehensive changes to the original work plans for a given day.

8.3 Alternative usage areas

Snow clearing is naturally not the only area where fleet management may be used (nor the most obvious). This section will present a few areas where such a software is also likely to be useful.

Fleet of salesmen

As mentioned earlier in the text, the VRP is basically an extension of the TSP to many salesmen. As such, a fleet of salesmen is a natural area of use for a fleet management system. Consider that a company has a fleet of salesmen who are to visit a number of cities to sell their product. Each salesman has the same goods capacity, while each city has its own demand. This problem is essentially the same as the problem the proposed solution is related to, with one difference: While the capacity in the snow clearing case is based on time consumption, the salesmen case has to do with capacity limitation. This is but a data set difference however, and does not require advanced reengineering.

Fleet of ships

When people hear the word fleet, they are most likely thinking about ships. And ships can definitely utilize a fleet management solution such as the proposed solution. The problem would look very similar to the salesmen formulation, and the big difference would be in the distance calculation, which now would be based on sea maps.

General areas

Generally, the VRP, and thus fleet management, is relevant for the following problems ([35]):

- Freight routing. Routing of goods. Both problems presented in this section belongs in this category.
- Service routing. Routing of service equipment and personnel. The proposed solution for snow clearing is a service routing problem.
- Passenger routing. Routing to meet transportation needs. Airline routing is an example of such routing.

Most problems belonging to any of these categories may be helped by a solution such as the proposed system. The need for such a system must be evaluated for each problem instance though, as some problem, especially if small, may not need such an extensive tool.

The proposed solution should be adaptable to most VRPs with modifications. These modifications can, in most cases, be implemented quickly compared to the time needed for the original implementation. Updating the use of

maps, the GUIs (which will often just need renaming of some fields) and the database, should not be a very time consuming process when the framework is already in place.

8.4 Distinctive Solution Characteristics

Since fleet management systems already are in widespread use, it is interesting to present a view on what makes the proposed solution stand out, if anything. Many fleet management systems, such as [36], delivers a combined hardware and software solution. One of the goals with the proposed solution was to make it independent of underlying hardware technology. All the system demands, with respect to other software and hardware the proposed solution requires with the current desing is: Java support and a GPS receiver. In addition the field application is made to work on an advanced, mobile unit such as i.e a PDA. (Applying it to i.e. a laptop in the truck would only require reengineering the GUI, the rest of the system should work correctly.) While this does not put the proposed solution apart from every other solution (there exist a lot of pure software fleet management solutions), it could at least make it interesting to those prefering such flexibility. (How flexible other software solutions are, with regards to i.e. operating system, hardware, etc. is hard to determine, as this information is not always publicly available. The proposed system should however run on all Java-supporting operating systems, making it highly flexible with regards to surrounding technology.)

As fleet management solutions already exist in abundance, it is hard to create a completely unique solution. Maybe the simplicity and limited list of functions can actually be positive, as it could offer some companies what exactly what they need, without having to by unwanted features.

8.5 Future Work

The main thing that needs to be done is of course completing the solution. While some functionality is already in place, there is certainly a lot of work left. Addressing some of the problems that have not been solved, (such as the continuous optimisation case,), will require some design effort, while i.e communication requirements should be clarified already and only need implementing.

The two GIS parts should presumably be the largest tasks remaining. Constructing these require more in-depth knowledge about how a GIS really works, and gaining this knowledge may be just as demanding as the following implementation.

A good idea for pursuance of this thesis is to work more towards the target group. While this text presents some of the demands for proposed solution seen from an ”outsider’s” point of view, contacting snow clearing companies is likely to increase developers’ understanding of what the system must contain.

Chapter 9

Conclusion

This report has tried to present some of the important aspects that need consideration when creating a fleet management system. While there are certainly options that have not been covered, some of the most vital aspects should be thoroughly presented. The solution was presented based on a given scenario, and though this scenario may be somewhat limited with regards to real life demand, the tasks carried out should still be the same in principle.

The optimisation theory imply the complexity of vehicle routing, shown by i.e. the fact that fourty year old algorithms are still in widespread use and only outperformed by much more complex solutions. In this report much of the focus has been on the heuristics that can be applied to the optimisation process, and while they give no accuracy guaranties they are well suited for the system in terms of speed. After testing the performance of the optimisation process using several approaches, the algorithm by Clarke & Wright ended up being proposed for the system. Its relatively good performance, easy implementation and processing speed make it ideal for the atleast-once-a-day optimisation process stated in the scenario descrption. Some advanced approaches could certainly give more accurate solutions, but they cannot match the speed of CW, which has been considered as important in this case.

For the localisation services needed the positioning itself should be based on GPS, but a system like the European Galileo project will be interesting once finished. For user interactivity a solution using GIS is proposed. Since developing such applications is a large task, implementations have been left out. The sections about GIS shows though, that with detailed maps, accurate road data, and continuously updated truck positions, a good GIS solution should be able to cover all graphical user requirements.

When designing the system, the focus has been on simplicity. The GUIs created should be easy to use, and generally include only two tasks; write and click. The system is designed in Java, J2SE in the central application and J2ME in the field application. This means that the application should be available for a lot of different systems, since Java is platform independent. The solution is also completely software based, making it independent of underlying technology.

A lot of work, such as implementing the GIS functions, still remain for the proposed system to be complete, and implementing the remaining functionality in co-operation with a possible customer could be a good approach to discover necessary functionality that has not been detected in the work process up until now.

References

[1] Trådløse Trondheim. http://www.tradlosetrondheim.no/. Last visited: 15.1.06.

[2] M. Rönnqvist. Traveling Salesman Problem (TSP). Lecture notes TIØ4150, NTNU, 2006.

[3] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970.

[4] G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation*. GERAD 25th anniversary series 5. Springer, 2005.

[5] D. Jungnickel. *Graphs, Networks and Algorithms*, volume 5. Springer, 1999.

[6] M. Kubo. Traveling salesman problem. Lecture notes, Tokyo University of Marine Science and Technology. www.tosho-u.ac.jp/~kubo/presen/tsp.ppt.

[7] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the travelling salesman problem. *Operations Research*, 21(2):498–516, 1973.

[8] M. Rönnqvist. Vehicle routing problems (VRP). Lecture notes TIØ4150, NTNU, 2006.

[9] M. Rönnqvist. Set partitioning and set covering models. Lecture notes TIØ4150, NTNU, 2006.

[10] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[11] J.-F. Cordeaux, M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53:512–522, 2002.

[12] M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124, 1981.

[13] B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974.

[14] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System - Theory and Practice*. Springer, fourth edition, 1997.

[15] C. C. Counselman III and I. I. Shapiro. Miniature interferometer terminals for earth surveying. *Journal of Geodesy*, 53(2):139–163, 1979.

[16] What is GIS? GIS.com - the Guide to Geographic Information Systems, http://www.gis.com/whatisgis/index.cfm. Last visited: 3.1.07.

[17] The Geodatabase View. ESRI - GIS and Mapping Software, http://www.esri.com/software/arcgis/concepts/gis-data.html. Last visited: 3.1.07.

[18] The Geovisualization View. ESRI - GIS and Mapping Software, http://www.esri.com/software/arcgis/concepts/geovisualization.html. Last visited: 5.1.07.

[19] The Geoprocessing View. ESRI - GIS and Mapping Software, http://www.esri.com/software/arcgis/concepts/geoprocessing.html. Last visited: 5.1.07.

[20] R. F. Tomlinson. *Thinking about GIS: geographical information system planning for managers*. ESRI Press, 2003. Excerpt of the contents of this book available at: [37].

[21] J. Benedicto, S. E. Dinwiddy, G. Gatti, R. Lucas, and M. Lugert. GALILEO: Satelite System Design and Technology Developments. Technical report, European Space Agency, 2000. http://esamultimedia.esa.int/docs/galileo_world_paper_Dec_2000.pdf.

[22] What is Galileo? ESA, http://www.esa.int/esaNA/GGGMX650NDC_galileo_0.html. Last visited: 6.1.07.

[23] How to build up a constellation of 30 navigation satellites. ESA, http://www.esa.int/esaNA/ESAAZZ6708D_galileo_0.html.

[24] First Galileo satellite on orbit to demonstrate key technologies. ESA, Press Release nr 61-2005. http://www.esa.int/esaCP/Pr_61_2005_p_EN.html. Last visited: 6.1.2007.

[25] Location Technologies for GSM, GPRS and UMTS Networks. White Paper, SnapTrack, 2003. Available at: http://www.cdmatech.com/download_library/pdf/location_tech_wp_1-03.pdf (Requires free TechRepublic membership.).

[26] P. J. Duffett-Smith and M. D. Macnaughtan. Precise UE positioning in UMTS using cumulative virtual blanking. *3G Mobile Communication Tecgnologies*, (489):355–359, 2002.

[27] Statens Kartverk. http://www.statkart.no.

[28] Statens Vegvesen. http://www.vegvesen.no.

[29] Java.sun.com - The Source for Java Developers. http://java.sun.com.

[30] NetBeans IDE. http://www.netbeans.org/products/ide/.

[31] Apache Derby. http://db.apache.org/derby/.

[32] Apache Software Foundation. *Derby Reference Manual*, 2007. http://db.apache.org/derby/docs/dev/ref/ref-single.html.

[33] ArcGIS – The Complete Geographic Information System. http://www.esri.com/software/arcgis/index.html. Last visited: 11.1.07.

[34] A. Osterwalder. *The Business Model Ontology - a proposition in a design sience approach*. PhD thesis, University of Lausanne, 2004. http://www.businessmodeldesign.com/publications/The%20Business%20Model%20Ontology%20a%20proposition%20in%20a%20design%20science%20approach.pdf.

[35] Martin Savelsbergh. Vehicle Routing and Scheduling. Technical report, The Logistics Institute, Georgia Institute of Technology. Lecture notes. http://www.ima.umn.edu/talks/workshops/9-9-13.2002/savelsbergh/VRP_part1.pdf. Last visited: 12.1.07.

[36] Starcom Systems – Logistics and Information Management. http://www.starcomsystems.com/.

[37] Implementing GIS. GIS.com - the Guide to Geographic Information Systems, http://www.gis.com/implementing_gis/index.html. Last visited: 5.1.07.

Appendix A

Use Case Diagram

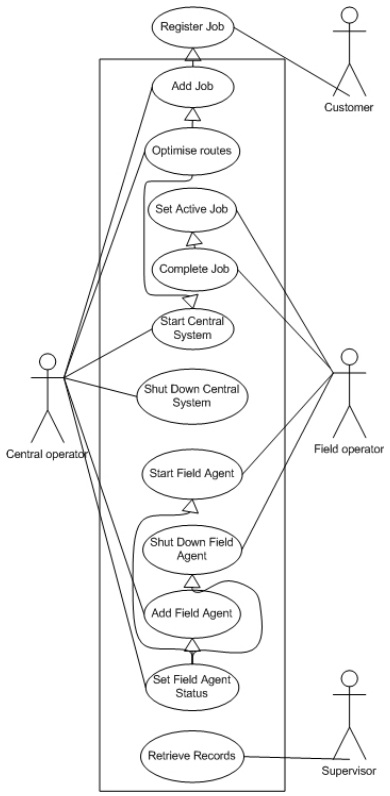

Figure A.1: Use Case Diagram. The rectangle represents the system boundaries.

Appendix B

UML Class Diagrams

B.1 Field Core

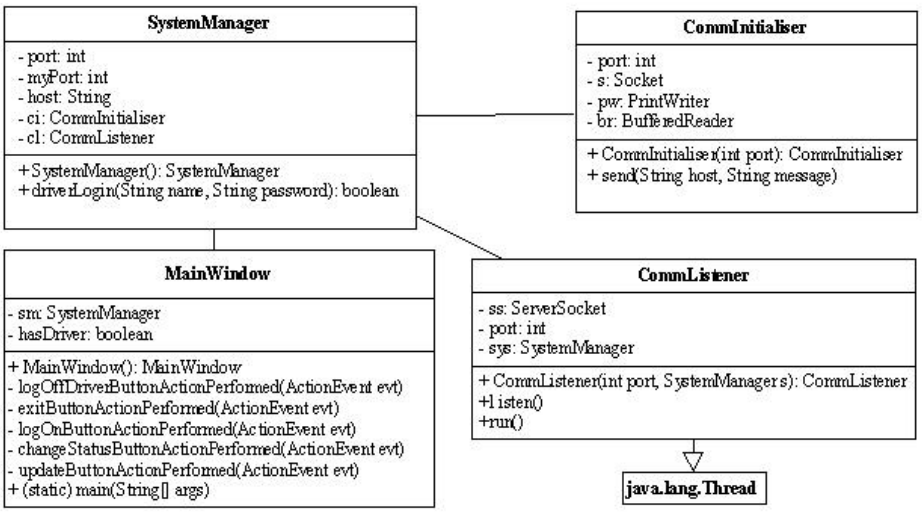

Figure B.1: Class Diagram – Package CoreSys (field)

B.2 Visual

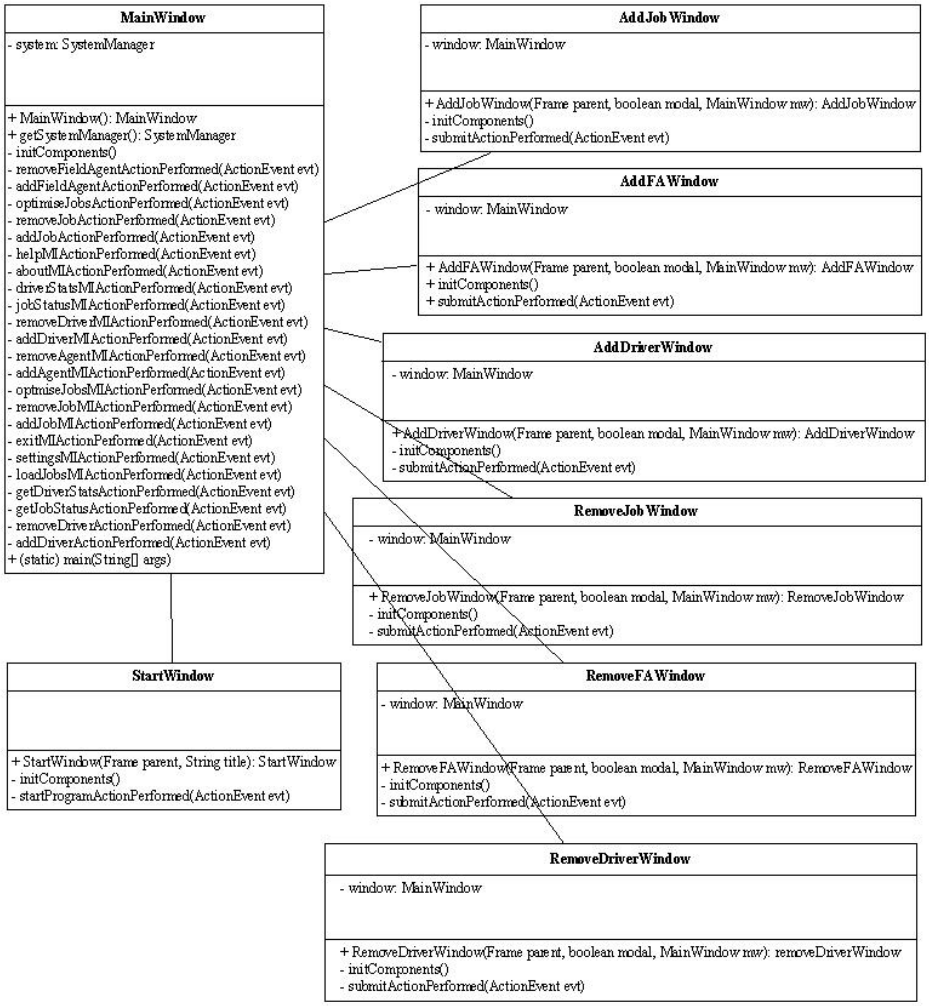

Figure B.2: Class Diagram – Package Visual

B.3 Central Core

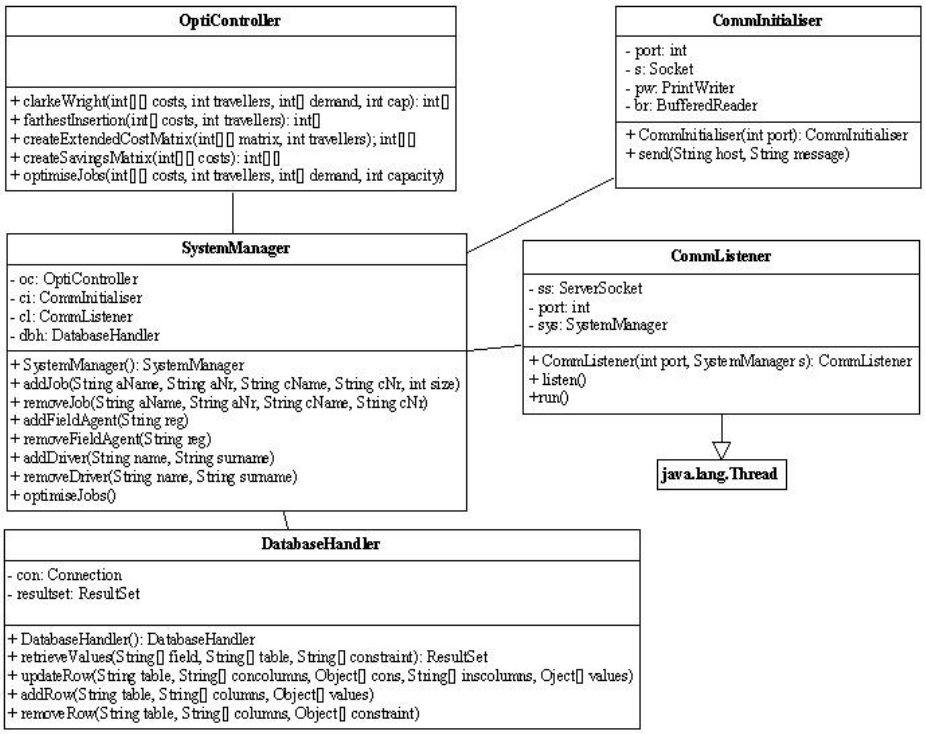

Figure B.3: Class Diagram – Package CoreSys (central)