CHALMERS

DEVELOPMENT OF DISPLAY TO OPTIMIZE THE FUEL FOR TRANSPORTATION FLEET

Master of Science Thesis SOFTWARE ENGINEERING AND TECHNOLOGY

Lars Larsson

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, November 2011

Development of display to optimize the fuel for transportation fleet

Lars Larsson

Examiner: Josef Svenningsson

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

Department of Computer Science and Engineering
Göteborg, Sweden November 2011

Abstract

As the fuel price is rising and the climate debate about global warming has taken off, the transportation sector has been forced to take action to reduce their usage of fuel. An important step to meet these new demands to become more fuel-efficient is to develop tools to help analyzing current fuel usage and find improvements that can be made to lower consumption.

This thesis focuses on public transportation where many vehicles have Intelligent Transport System (ITS) installed, and where a small improvement of a few percentage can give a large benefit. In this thesis I will propose some ideas on how to use the data collected by the ITS to support bus-operators in their effort to improve efficiency. Further, the thesis will present a prototype application that implements a few of these ideas.

The developed prototype can be used to summarize the collected information for different entities allowing insight about driving behavior and comparison of the entities such as drivers and vehicles. It will also elaborate on how to present such summaries visually and how to work with large dataset while trying to keep it user-friendly.

Keywords: Busses, public transportation, fuel consumption, fuel-efficiency, Silverlight, human-computer interaction

Table of Contents

Abbreviations

AIC
: Advanced Information Center is a device placed inside a bus to transmit data to the central system and provide information to the driver.

API
: Application Programming Interface defines and describes how to write an application that uses a functionality of an operating system or a program.

CAN
: Controller Area Network is an internal serial bus often found in automotives.

FMS
: Fleet Management System is a common protocol supported by many vendors. But it can also refer to applications built on this protocol to allow monitoring and managing of a large vehicle fleet.

GPS
: Global Positioning System used to position e.g. a bus in the real world and on a map.

ITS
: Intelligent Transport System could be described as computer aid for the transportation sector.

Journey
: Journey is associated with a route and a time of the day.

Odometer
: Total distance traveled for a vehicle.

Route
: Defines the roads and stops the vehicle should travel.

XML
: Extensible Markup Language is a format for packaging data similar to that of an envelope.

1 Introduction

The transportation sector has undergone a major change the last decade with the introduction of Information Technology (IT) such as Intelligent Transport Solutions (ITS). In recent years, the transportation sector has started using ITS solutions as a means to cut the total fuel consumption for their fleets. The importance of fuel consumption has increased both due to the economic situation where oil prices are increasing and due to the political aspects and debate about global warming

The focus of this thesis has been in the public transportation sector using ITS to assist in lowering the fuel consumption and monitoring the usage. The public transportations system is composed of buses, trams, ferries and trains, to name a few. All of these need energy either in form of fuel like gasoline, diesel or one of many carbon dioxide (CO_2) neutral alternatives. Even energy source like electricity that is often considered to be a green energy source may be the result of coal or oil combustion at power plants or in some other way affecting the total CO_2 emissions. Thus, it is important to try to make the use of these energy sources more efficient. Another important reason is that these resources are expensive and a more efficient use will save a lot of money. An example is that a very small improvement of a few percentages means a reduction up to thousands of liters of gasoline per bus and the possibility for saving millions of euro for a larger fleet. This money can then be used to reduce ticket prices or increase the number of journeys and thereby persuade more people to leave their car at home and use the public transportation creating a positive chain of events.

Considering that the public transportation is politically influenced, the bus industry has become pressured into cutting its CO_2 emission in all parts of their organization. To do this they need become more efficient in all parts, from the bus depot and the office down to individual busses[1]. This pressure comes from the fact [2] that 20% of the European Union CO_2 emission comes from road transportations, so getting people to leave their car and take the bus becomes increasingly important if EU is to achieve its goal of lowering its CO_2 emission by 20% before 2020. To make people leave their car an important factor is having a comfortable ride and this is affected by the driving behavior and can be improved by embracing eco-driving principles. Other political arguments involve security and geopolitical reasons such as reducing oil dependency and transition to green-technology. But perhaps a more important and local reason is improvement of air quality and noise level within a city.

1.1 Purpose

The task of this master's thesis project is to develop an application to show statistics of fuel usage for busses. The application is a tool used for the central management to identifying existing inefficiencies. The work includes developing ideas, designing and implementing a subset of these ideas in the application.

The project is built to use an existing ITS framework provided by Consat Telematics for data gathering from the vehicles. As part of the optimization the solution should preferable, allow monitoring the fuel used within the organization and allowing for comparison of different entities and settings. The result is to find improvements that can lower emissions and save money on fuel expenses.

The task includes analyzing the available parameters and the relation to the fuel consumption, as this is often not known by the operators or the manufacturers of the vehicles. Furthermore, a subset of the parameters will be selected to be included in the application and the creation of views to display the parameters in order to support decisions and changes in the application.

It has also been proposed to optionally make the application automatically identify improvements. These could then be ordered by their significance coupled with a detailed description and data it was based on.
Example of scenarios that could be searched for is:

- If busses are standing still with their engines running
- If drivers are accelerating rapidly or harsh breaking as these causes unnecessary fuel usage
- Best vehicle for a given route

A possible outcome of the improvements found could be to instruct and train underperforming drivers in eco driving, or let it affect their salary. It could potentially also be used in the planning of the routes and roads to allow a better flow of traffic, and creating a more accurate time schedule. An important aspect to this is to analyze the result before and after a change to be able to draw conclusions about its effect. This can be done due to the repetitive nature of public transportation.

There is existing commercial-off-the-shelf software for data analysis and visualization such as *spotfire* and one solution would be to export data and integrate with these third party APIs. However this would create an external dependency on such tools and would make it hard to use and integrate with existing platform.

1.2 Goal

The goal consists of identifying important parameters and factors for fuel consumption in busses by studying previous work in this field. Furthermore, the goal is to development a prototype application in Silverlight 4 with different views for visualizing the data of the parameters identified. The application is aimed to assist the public transportation sector in lowering their fuel consumption.

Furthermore, a set of search criteria is to be defined for the application to limit the dataset and to allow for a quick response time. The search criteria will be used to ask questions to the system for example how hybrid busses perform on a route compared to non-hybrid busses. The result should thus be able to reflect this.

The application shall be compatible with the current databases deployed by the ITS framework and get its data from there. When the needed data is non-existing in the database or the response time is to slow, a table for this project will be added to the ITS framework database. The table will utilize the framework in importing data to the tables.

The application shall allow for summarizing the data for several different trips and be presented in a graph with different levels of detail according to the search criteria given by the user.

The application shall allow for comparison of different settings and parameters such as

- journeys
- vehicles
- drivers

1.3 Delimitations

The project is intended as a prototype and not a commercially ready product. This also means only one culture and language will be supported, which will be English.

The program part regarding comparing configurations and parameters will be focused and limited to three aspects: the vehicle, the drivers and journeys. The search parameters will be based on these.

There will also be limitations to the accuracy and correctness of the logical mapping due to measurement errors and unplanned scenarios. For example a bus that does not follow its planned route for some reason.

Automatically detection of inefficiencies, as proposed in the Purpose section 1.1, will not be implemented but instead the application will allow for an overview that could fill the gap.

Due to limitations of the system, there will not be any real-time feedback to the drivers such as instantaneous fuel usage.

2 Background

This section is intended to give some background of the domains relevant for the project. That includes public transportation organizational overview, relevant stakeholder and existing technology used.

2.1 Organizational Overview

The organization for managing public transportation can differ a lot between different places but generally one can describe it as having one transport authority (a region or city) that has one or more operators. An operator can be the same as the transport authority, founded and owned by the region or it could be a commercial bus company. The main objective of the transport authority is to guarantee regular transportation that is comfortable, on time and costs as little as possible. In other words, they wish to ensure that the operators fulfill their obligations and contracts. Since the transport authority is a government institution, they can affect the environment and conditions for a route such as the planning of the routes. This gives them the ability to decide what the roads should look like, if there should be a public transportation lane, how the red lights should behave to give the best flow. To their aid, they have advanced tools to assist them in planning these routes and their schedules. However these tools fails to capture and assist in considering unforeseen delays such as red light, road crossings and how they affect the travel time.

As mentioned above there are also the operators for the routes. Their main objective is to maximize their profit. Achieving this goal of theirs means they have to perform their obligations well while keeping down their costs. It is thus vital to maximize the efficiency of their resources as to cut costs. Therefore, this actor is the one most concerned about having high fuel efficiency and it has the most to gain, since high efficiency can be turned into profit.

Maximizing the efficiency for an operator includes many aspects where one important is to use the right bus for the right job. What the right bus mean is usually to have the smallest bus as possible for each journey and still fit all passengers. The reason for this is that a small bus cost less and has lower fuel consumption, due to its lower weight. But there are also other factors to consider such as the characteristics of the buses. The characteristics can be the engine type and configuration of the gears. The engine type can be diesel or hybrid to name a few. While an example of different gear configuration could be that the operator uses one configuration for routes within urban areas and another configuration for the gears outside urban areas.

In summary, the following stakeholders can be found for the domain:

Transport authority – City or region that buys a service for their citizens and guarantee good service
Traffic planners – Concerned about the route, roads, traffic lights and keeping a good traffic flow
Operators – Operates the busses and want to minimize costs and fuel
Drivers – Is employed by an operator to perform journeys with a vehicle

2.2 Platform & Infrastructure

Intelligent Transport System (ITS) [3] has become a general term for integrating Information Technology (IT) with the transportation sector. That is adding information and communication capabilities to the vehicles as to improve the safety, efficiency and simplicity. Examples of what this could be in practice regarding public transportations include:

- Keeping track of what to do next, for example what route to take and when to turn etc.
- Monitoring vehicles location in relation to its schedule
- Communication with a central location, such as position and encountered problems

Consat Telematics is a sub company to Consat and is working with information management for the public transportation sector. They have together with Volvo Bus Corporation created an application called ITS4mobility [4] which is an ITS-solution providing functionality with respect to:

- RTPI (Real Time Passenger Information)
- Traffic monitoring and planning
- Fleet Management

The system consists of several separate parts including:

- vehicle system
  The vehicle system is composed of a computer with some modules that are attached inside the vehicle and are referred to as Advanced Information Centre.
- communication system
  The communication system is used for relaying information and data between the vehicle and central system, using GPRS.
- a central system
  Receiving and storing vehicle data that can later be viewed by applications.
- at-stop displays
  Displays located at the stops to inform passengers.
- central system applications
  Desktop application for interacting and displaying the data stored in the central system.
- depot server
  Used to upload data logs from the vehicles and store it in the central system.
  It also provides updates for software and updates of the vehicle configurations.

The system thus interacts with different types of users and stakeholders including

- Drivers
  Interact with the vehicle computer.
- Traffic controllers
  Monitors the real-time traffic situation and can change the traffic or communicate with the drivers.
- Traffic planners
  Uses statistics and history to plan and improve future traffic.
- Passengers
  Using the busses for transportations and is given information e.g. forecasts about the traffic situation.

Responsible for the communication and interaction with the driver in the vehicle is the Advanced Information Centre (AIC). The AIC contains GPS receiver for absolute positioning and a GSM/GPRS modem for communication. It supports several different manufacturers communication standards but most important is the "*Bus FMS-standard*" [5]. The "*Bus FMS-standard*" is an industrial communication standard that defines a bus gateway which regularly announces the value of certain defined parameters from the internal OEM specific bus. This standard allows the reading of current speed, acceleration, rpm, brake, gear, parking brakes, traveled distance, door opened, fuel consumption, engine hours to name a few of the available parameters. On top of this comes some non-standardized parameter that depends on the manufacturer. Only a small portion of this large mountain of data is sent in real-time to the central system. Some additional data is logged into files that are later uploaded to the depot but it is far from everything.

The system uses the real-time data and history of previous vehicles to make forecasts of remaining time until arrival. This information can then be broadcast to the affected stops as to inform any waiting passengers. It also supports sending out custom information like disruptions or accidents to

the affected stops. Furthermore, the system automatically announces next-stop for passengers onboard the vehicle and final destination for people outside the vehicle.

Additionally the system allows the traffic controller to keep track of each individual vehicle using both its geographically position as well as its logical route position. To clarify, the logical route position allows the planner to quickly see how the vehicle is progressing in accordance to its route. This data allows the system to estimate the time deviation from the timetable that can be used to inform any waiting passengers as described earlier.

Another application that they deliver which is more closely related to this work, is their Fleet Management system[6] also known as FMS. The fleet management system is about managing the vehicles and presenting summaries of the data collected from each individual vehicle. With this application, one can view data for individual buses or collection of busses in a spreadsheet. This gives a powerful tool to support decisions and following up changes.

Another actor on the market and main competition is the Sweden based Hogia [7] which delivers PubTrans, a centralized system for data storage target to the public transportation. Based on this system they offer components and subsystems that can deliver travel information, ticket handling, travel planning, traffic control, vehicle report system etc. In the PubTrans they save huge amount of data to allow following up on events and creating statistics.

3 Methodology

Due to open nature of the problem, an agile process with its dynamic and flexible nature seemed most appropriate. A reason for this approach was to allow experimenting and an iterative approach where I could meet and discuss how to continue the project at regular intervals or milestones. As I had a limited knowledge in the area it was important to stay in close contact with one of sales-representatives. This was vital as to get an insight about what potential customer would expect and find useful. Some examples of things that were discussed include new problems, the system look-and-feel, workflow and visualization of the result. All the work for this project was performed at Consat as this allowed me to receive feedback face-to-face and ask questions directly to their developers.

The project started with the gathering of information about how to identify the inefficiencies of fuel consumption and background research. After that, I created some initial ideas of how the results could look visually without considering limitations too much. At the same time, I was familiarizing myself with the system and technology.

The implementation was carried out iteratively with small upgrades on the prototype as to reduce complexity and having to avoid breaking the application for long duration of times. The application was also initially divided into several different sub applications or views to further split the problem and concerns. As previously mentioned there was quite a bit of experimenting where one such approach was to use excel to plot data collected directly from the database to see if one could draw any conclusions from the resulting graph. Another experimenting involved the GUI and workflow as I lacked the knowledge in designing an easy to use interface. Most of which had to be discarded in the end.

The final stage of the development was spent on doing a final integrated application with the most interesting views developed.

4 Pre-study

This chapter is aimed as an initial analysis to determine parameters and configurations effect on fuel consumption and is the background material for the choices made later in the project. It also aims to review related work and find what affects fuel consumption.

4.1 Parameters

There are relatively few studies on busses and their fuel efficiency. But considering that a bus is closely related to a truck as they often share the same chassis but with different bodies, they should have similar behaviors.

There is however a relatively old study[8] on busses and the importance of certain parameters where they try to estimate its affect and allow for calculating consumption.
In the first phase of the study, they reviewed:

- Bus characteristics
- Usage and route
- Driver behavior
- Road characteristics

Some of the conclusions made were that the bus model and route were the main factor to explain the differences in fuel consumption. They found that the fuel consumption for the least efficient combination and most efficient combination of route and bus were 2.17 l/km and 0.84 l/km, which signifies a big difference in operating costs. They also found that the time on the day performing the route was not relevant in their case and rationalized it to be due to bus-only lanes, thus not affected by rush hours. Furthermore, they concluded that the age and total mileage of a bus seemed to have only little impact on the fuel consumption.

In the second phase of the study another set of parameters were studied such as:

- Weight from passengers
- Travel speed
- Stops and obstacles for the route

The result was that they all had an important impact were the speed and number of stops per distance accounted for two-thirds of the variation.

Another study[9] showing the energy losses for heavy trucks weighing 40 tons (with load) on highway driving. In the simulation it was found that about 40% of the energy produced by the engine is lost due to climbing uphill, 30% as rolling resistance at the wheels, 23% is lost due to air resistance and only 7% is lost to internal mechanics such as gears, engine auxiliary unit etc. It was also concluded that the conversion from fuel to energy only had 40% efficiency rate meaning that an entire 60% of the energy is lost as heat inside the engine. However, the simulation weight of 40 tons is over the expected value of a bus with passengers and highway driving is not comparable to city traffic. But it does give an indication of the numbers even though breaking can be expected to be a lot higher for a bus in city traffic.

4.2 Eco-driving

Eco-driving also known as driving eco-friendly is about driving a vehicle in an effective manner and thereby lowering fuel consumption and emissions. This doesn't mean using CO2 neutral fuel (green fuel) but about lowering the effect of using fuels such as gasoline or diesel. It is an important

part if EU is to reach its goal of lowering CO_2 emissions by 20% before 2020. This should be seen in the light of "*EC legislation to recduce emission through 'complimentary measures' which could be gear-shift indicators, tyre pressure monitor (TPMS), low energy tyres*"[2]. It could be summarized as making the driver aware and a part of the effort to lower the fuel consumption by providing information to the driver. It should be seen as compliment to the more expensive technological solutions.

These complimentary measures can often be tied back to basic rules of eco-driving. These rules are outlined, justified and explained in details in *ECODRIVEN Campaign Catalogue for European Ecodriving & Traffic Safety Campaigns* [2] and these can be summarized as

1. Use the highest gear possible
   Change to a high gear on relatively low rpm as higher gears are more effective.
2. Maintaining a steady speed
   Anticipating traffic flow and obstacles as to avoid heavy acceleration and hash breaking.
3. Avoiding high speed
   Speed outside the range of 50 km/h to 90 km/h has a great negative effect on emissions.
4. Don't idle
   Don't let the engine running while not in use for long periods.
5. Reduce resistance
   Check tire pressure and remove unnecessary windbreaks.

4.2.1 Projects

European Intelligent Energy Europe (IEE) has had several programs such as "*Training Programme for Local Energy Agencies & Actors in Transport & Sustainable Energy Actions*" known as TREATISE and "*European Campaign On improving DRIVing behaviour, ENergy-efficiency and traffic safety*" known as ECODRIVEN [2].

ECODRIVEN[2] was aimed at passenger cars, delivery vans, lorries and buses. The project spanned across 9 different EU countries and it was conducted together with a local partner. The conclusion of the study was that eco-driving is simple yet powerful way to reduce CO_2 emission and fuel consumption. By comparing the costs of promotion activities and training of the drivers to its benefits, they found it to be very cost-effective. In the ECODRIVEN project, the reduction immediately after training was as high as 15-20% and around 10% as long-term saving.

Another benefit they found in ECODRIVEN was that the emission rate and fuel was coupled with accident rates and safety. In certain cases like "Hamburger Wasserwerke" that trained its 91 drivers, they experience a 22% decrease in insurance claims as result of the training. This was contributed to the planning for the road ahead by avoiding high speed and calmer driving that eco-driving emphasizes.

Finally, the servicing and repairs needed on the vehicles was also reduced as the calmer driving tear less on the different parts of the vehicles like the brakes and tires. Even the travel time for a journey maintained the same time or in certain cases was slightly improved so the conclusion was that they were unable to find any drawbacks with eco-driving.

Another project [10] conducting an eco-driving pilot program for bus drivers in Athens, Greece, found that by comparing the fuel consumption before and after the training, that the company Thermal Bus Company (ETHEL) saved in average 4.35% in fuel consumption. This gave them an annual saving of €1697 per bus, considering their average consumption for a bus were 60,000 liters per year and that the fuel price was €0.65 per liter. Furthermore, this would give a total annual saving of €2,994,900 for their entire fleet consisting of over 1700 busses. This surpassed the cost of the training

4.3 Monitoring and Feedback Projects

There have been numerous projects and studies a on how to lower the fuel consumption, across many countries. One of these was shown in the Euro Bus Expo brochure for 2010[1] stating that: *"Research has found that GreenRoad Safety Center™ can reduce accident rates by more than 50% and reduce fuel costs and CO_2 emissions by an average of 7%. The system improves driver behaviour behind the wheel through an in-vehicle sensor monitoring up to 120 driving manoeuvres, such as speed, braking, acceleration, lane handling and turning."*

And another project shown in an article in a Swedish journal Forskning [11] for January 2010 about *Intelligent Speed Adaptation* (ISA)[12] that is a device that is mounted inside a car and uses a GPS-receiver together with an internal control unit to determine the current speed and the maximum allowed speed for the road. If driving faster than allowed the device will give off a beep and a text is displayed to notify the driver that he or she is driving too fast. This was installed in many of the vehicles possessed by the region of Skåne in Sweden, as a co-operative effort between the region and the road department (sv. Vägverket). The result was that given this feedback to the driver would save 9 to 13% on fuel consumptions which would give an annual saving of tens of thousands of euro.

5 Technologies

This chapter is dedicated to technical knowledge which was considered before and during the project. These are a pre-required knowledge if one is to fully understand the considerations and choices done during the "Design & Result" chapter. The information about the technologies is a compilation of a variety of sources.

5.1 Database

As stated in the Goal section 1.2, the project will be based on Consat Telematics framework and use its databases as much as possible. Therefore, an introduction on the database layout and available methods for accessing it will be outlined.

Their existing system and framework uses several databases with data and information in a standard relational database, Microsoft SQL. It is divided such that one database is used for organizational information such as routes, journeys, traffic planning etc. As for the gathered data and statistics, they divide it into several databases depending on application and usage. These statistics databases are often organized to match the data needed to generate a specific report. That is one entry can be seen as a summary of data and is composed from many logged data rows. This allows having a low number of rows and entries in the table while also keeping down the response time. Consequently, whenever data is received from a vehicle that matches a database entry, that entry needs to be updated to include the data.

As for accessing the database there are several options: one can either have stored procedures in one of the databases that collect data from one or several tables, local or remote and give back a complete report for the given in parameters. The advantage with this is bandwidth effectiveness & safety while there is some additional work in creating the procedure. The disadvantage is that the database can become a bottleneck if the procedures are processing intensive. Using stored procedure is how the existing reports are generated within their current product, but using services that accesses and caches data to reduce load.

Another option is to use a Data Access Object (DAO) which is an object that provides an interface to the database but from the programmers point-of-view looks like a normal object as it hides this connection to the persistence layer. In C# and .NET there is a standard persistency layer called LINQ that provides the functionality of DAO objects where there are no programmer-visible distinctions between objects connected to a database or local objects except that somewhere they are coupled with different data providers. There are some restrictions to this such as how data can be moved between providers. Another neat feature with LINQ is that it allows one to do object queries similar to that of SQL to both local and remote collection of objects. As for a database provider there is some additional work when creating a class such as add metadata to properties and methods to instruct LINQ to map the local resource to that of a remote resource. This is also known as Object Relation Mapping (ORM). An example of what the metadata could contain is table information, the column that a property should bind to, how to convert the data from the remote to the local etc. Beyond the basic property mapping, one can also have procedures in the class mapped to a stored procedure on the database server. So there are a lot of options but luckily most are not needed and it can be auto generated or set from within editors like visual studio.

There is also the most basic and primitive but well known method, where one write the SQL query directly inside of the code as a string and send it to the database. The resulting set is then manually traversed and insert into objects.

5.2 System Architecture

The system architecture is the model and structure that defines how the system operates. It is chosen during the design and before the implementation starts as it cannot be changed easily once the implementation starts. There are several different system architectures available to choose from. Each of these architectures has their advantages and disadvantages but all of them will have an important impact on the overall system and project. By analyzing and being aware of these one can better understand different problems and being able to find a suitable solution. Here are some common system architectures and patterns related to the project.

5.2.1 Client-server Architecture

The client-server architecture[13] is an umbrella term for any application that is divided into two or more distributed processes such as a client application communicating with a server application. Due to this wide definition, any application that uses a database can be considered a client-server application as long as the retrieval is located in the database process and data presentation or manipulation in the client process. The advantage with this approach is allowing several different views and users to work on the same dataset.

One of the most common architecture and what many people often associate as client-server architecture is a 2-tier architecture where the application or platform is divided into two logical parts. The advantage with this is that it is easy to reason and quick to implement while the disadvantage is that the solution often becomes adapted to a specific problem and not very general. The result is that a small change or extension could mean one have to rewrite the application.

5.2.2 Multi-tier Architecture (n-tier)

A multi-tier architecture also known as n-tier is another form of client-server architecture where the system is divided into several separate modules or layers where each layer has a defined task and interface to perform. Each layer should preferable only communicate with the its neighboring layers and thereby keeping a simple hierarchical architecture where one can easily add, remove or replace a layer with another implementation. The most common n-tier is the three-tier architecture [13] and involves dividing the system into three parts. It can be seen as an extension of the 2-tier architecture described earlier. The three layers are usually divided into:

- Data tier
  Usually consists of a database where the data is stored in a standardized and platform independent way.
- Logic tier
  It constrains all the business logic of the system such as functionality and data processing.
- Presentation tier
  It is responsible for presenting the functionality and data, to the end-user. Having this as a separate layer allows several different presentations, depending on the end-user and the required functionality from the logic tier.

This layout allows the different tiers to be more loosely coupled from each other and thus a greater flexibility for modification, replacement or changes to each layer. The connection between the layers is done by defining an interface for each layer and then uses a middleware for accessing the functionality. The use of middleware is to make this connection weaker and allow for greater flexibility such as allowing the layers to be located on the same computer or distributed on several computers.

This architecture has become the base for internet applications and many corporate applications as it allows data to be accessed and processed from a remote location by many users while keeping a flexible design. The disadvantage is the increased complexity and security concerns.

5.2.3 Service Oriented Architecture (SOA)

Service oriented architecture is a concept based on a distributed system which is location and provider independent. The concept focuses on the message communication [14] between the parties rather than on the implementation or environment surrounding the communication. This means that the protocol used to exchange these messages can be changed without affecting the service. Furthermore, this allows the service to be platform independent and works in non-homogenous environment. As for using a service, SOA defines an auto discovery mechanism that is based on metadata attached to the service, explaining what it does and how to use it (the semantics of the service).

5.2.4 Web Services

A web services (WS) is a concrete and commonly used way to implement a SOA service. A web service can be described as an operator independent API known as a service that is accessed and executed on a remote host, the provider of the service. Two important properties of a web service are that it is self-describing and discoverable. This allows for platform independence and simplifies integration between different applications.

There are however, two main types of web services. One based on remote procedure calls (RPC requests) which is process oriented and operates very much like any other software except that some code may be located on a remote machine. An alternative to RPC is RESTful web service which is resource-oriented and stateless. RESTful webservices works by exposing the direct underlying resource in a statelessness manner unlike RPC which uses states. RESTful is the basis for normal web sites where you access the resources by different URIs e.g. *http://baseruri.com/resource/95* to access an element of "*resource*" with identifier 95. This is opposed to RPC that uses an URIs for accessing the service and then uses messages to access a specific resource. Both types of web services can be used with a web protocol such as HTTP and SOAP for transportation. While the message data is sent usually is XML[14] documents.

5.2.4.1 WCF Services

Windows communication foundations (WCF for short)[15] is software development kit (SDK) for developing Web Services and is a part of Microsoft .NET platform. The metadata describing the WCF service is published using WSDL or another industrial standard. Using the published WSDL allows a client to automatically generate a proxy class for communication with service, using standard software tools. This proxy class then exposes the same functionality as the service and thus hides the underlying communication between client and server.

To set up and create a WCF service one needs to understand a few basic concepts that can be tied back to SOA. These concepts are endpoint, addresses, bindings and contracts.

Endpoint

An endpoint defines how to be able to reach and communicate with a service and consists of three parts: an address to identify the host, a binding to define how to communicate with the host and a contract which defines the service. Additionally a service may have several endpoints.

```
<endpoint
            address="http://localhost:8080/Service"
            binding="BasicHttpBinding"
            contract=" MyService"
/>
```

The example illustrates an endpoint listening on address *http://localhost:8080/Service* it communicates with *BasicHttpBinding* which is a configuration package defining communicating over http for the service and finally that the service provides or implements *MyService* interface.

Addresses

An address is a Unified Resource Identifier (URI) which defines the location of a service and protocol to use for transportation (transport schema). There are two types of URIs where the first type is an Uniform Resource Locator (URL) which can be used to access a resource e.g. *http://google.com*, An URL describes both how to communicate and where it is located. The other type of URIs are Unified Resource Names (URN), which is used to give a unique id to items and objects e.g. would be *urn:issn:1535-3613* which is used to uniquely identify a book. Usually the URI for WCF services are that of an URL and follows a formatting code as demonstrated here:
[transport/schema]://[host address][:port (optional)]

Bindings

Bindings are about defining the communication between the hosts such as what protocols to use, what security to use and what encoding or feature to activate. One such possible set of choices grouped together is what can be called a binding. An example is if one should use HTTP/HTTPS, TCP or another custom protocol for communication. As for encoding, it could be text, binary or using different compressions.

Contracts

Contracts are specification of how the service operates and how it can be used. There are several different types of contracts as listed below that help define the service.

Service contract
: What operations that can be performed.

Data contract
: What data to pass as parameters to an operation and returned by an operation.

Fault contract
: What errors that can be raised by the operation (exceptions).

Message contract
: Interactions directly on the message level instead of data level. Used to create an own message format and having control over the communication.

A simple example would be:

```
[ServiceContract]
interface MyService
{
      [OperationContract]
     List<Vehicle> GetAllVehicles();
}

[DataContract]
public class Vehicle
{
     [DataMember]
     public String Name { get; set; }
}
```

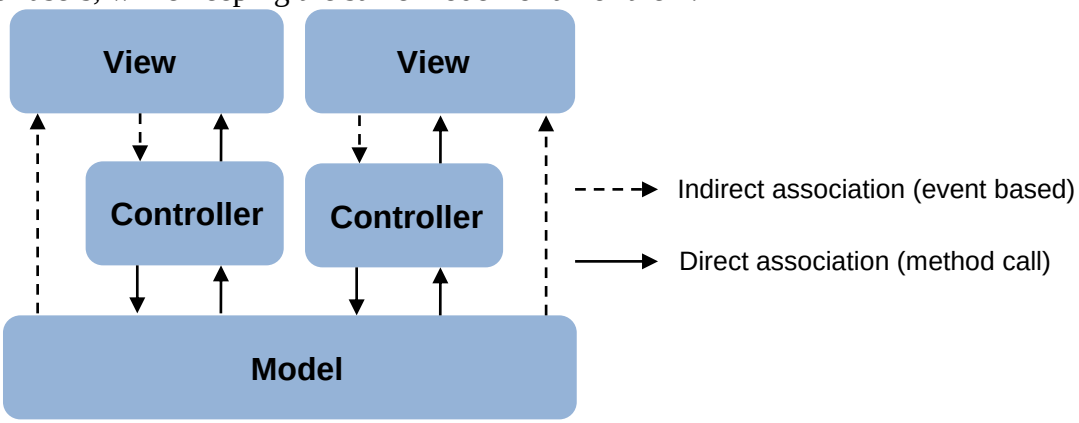
5.3 Application Architecture

An important part of the application is how the component themselves are designed and what general principle they follow. A recurring pattern or common structure simplifies programming and understanding as one can use pre-existing knowledge from one place in another. The direct consequences are shorter development time and reduction in the number of faults introduced into the application.

Another important task of application architecture is about dividing a problem into smaller tasks and creating logical components. This is fundamental if one is to allow for rearranging or replacing components and to simplify future development. The drawback is that it takes time to learn patterns and methodologies as well as to apply them on a problem. But it is often worthwhile and cost effective in a longer perspective.

5.3.1 Model-View-Controller

Model-View-Controller (MVC) [16][17] is about separating the user-interface from the underlying model, i.e. the application data and business logic. The most compelling reason to do this is to allow for redefining and changing the *view* of the application while keeping the same model. The other reason is that an application usually has several different stakeholders with different goals and needs. This is often satisfied by creating separate views with only the functionality needed for each group of users, while keeping the same model for all of them.



It should be noted that the term view is often a graphical interface but does not have to be, it could for example be an API interface to the model. As for the responsibilities, the view acts as a presentation filter of the functionality of the model. The controller acts as the mediator by handling events in the view and reflects this on the model and vice verse. Based on MVC there is Model-View-Presenter (MVP)[18] that decouples the state of the view into the controller and calls it a *presenter*. The presenter thus no longer needs to know the details of the view as it has local knowledge of its states. Tweaking this a little by moving the view states into a *presentation model* and one get what is known as Presentation Model (PM)[19].

5.3.2 Model-View-ViewModel

The Model-View-ViewModel (MVVM) [20] pattern can be seen as a special case of the Presentation Model pattern with support by the framework itself for synchronizing the visual state through what is known as bindings. A binding is a connection between a source and destination to allow data to propagate whenever the source is update. It is used in the MVVM to update widgets so that they reflect the *ViewModel* at all times. Unlike the presentation model, the ViewModel reflects the model in a way that is suitable for the view by getting and converting data as needed. This makes the ViewModel a mediator between the view and the model.

Not only does this pattern binds the displayed data but can also bind functions to react to actions and events by binding the widgets event/command handler to a function through the ViewModel.

Commands are capable of modifying the behavior of the controls and connect it to some business logic. An example of this is that a command can decide if a button can be pressed and what actions to take when the button is pressed. This pattern allows the view to only handle the layout and collection of the widgets as everything else is bound through the ViewModel.

5.4 Silverlight

Silverlight is a framework to build rich internet web applications [21] similar to that of Adobe Flash and Sun JavaFX. Just like Flash, it can be run in a web browser by installing a plug-in to the browser and is to some extent platform independent. It is not limited to the browser as it also supports running in a native window as a standard desktop application.

What really makes Silverlight stand out from the rest is its support for Microsoft .NET technology (including the CLR) and the use of Windows Presentation Foundation (WPF). The use of WPF allows for separation of graphical interface from the code, by defining the interface in what is called XAML. A XAML document is a XML document defining the layout and widgets of the interface and is thus loosely coupled to that of the code behind. The XAML document is however tied to a back-end class that hooks everything up and fills the gaps of things that can't be done in XAML.

5.5 Data Visualization

Data visualization is about how to present a large amount of data for humans to help them make decisions, see patterns or to make new discoveries. We use data visualization to help amplify our observations and understanding. When successful with this one achieves what sometimes is known as *cognitive amplification*. One process used to get this result is known as *knowledge crystallization* [22]. This process is about organizing and filtering the data in steps. Based on the information one creates a representational framework where one inserts the data collected. The framework is then used to filter out the data to smaller dataset and create a conclusion. The process can be described with the following steps:

1. Information gathering
   Collect and gather information about the problem.
2. Selecting information parameters
   Decide on the relevant parameters for the objective. It's about defining what attributes is important for the problem.
3. Organize information
   Organize the data using the selected parameters, e.g. create a table containing the selected parameters for the data.
4. Manipulate the information to problem
   Manipulate the data, reorder and reorganize. Remove data that is no longer relevant for the problem.
5. Processes the information (reduction)
   Group the data and reduce, summarize it and simplify.
6. Visualize/Output the data
   Visualize it and draw conclusions.

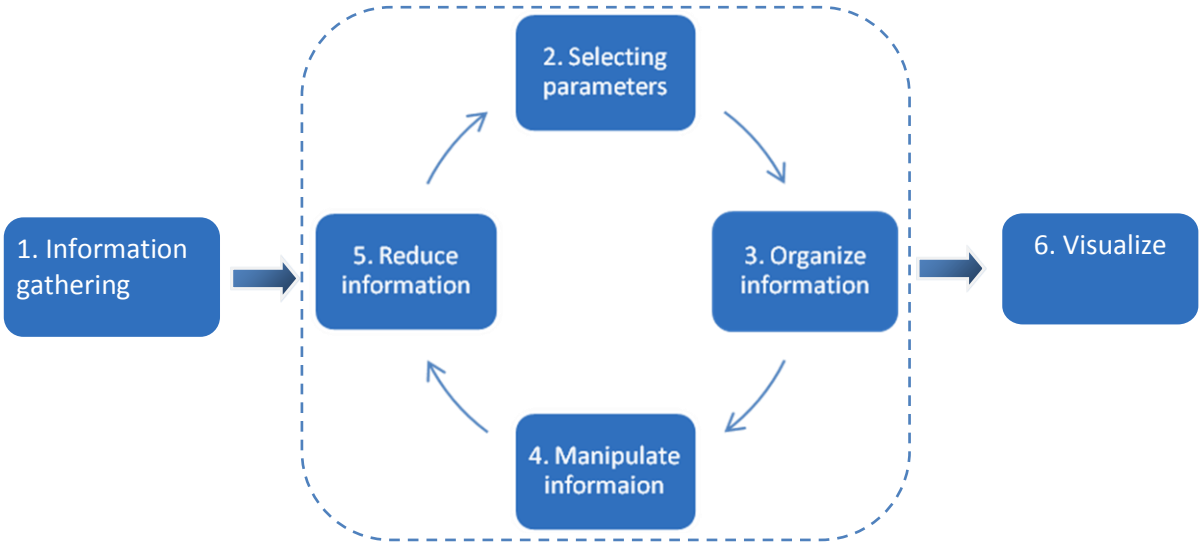


Figure 1: A simplified version of the *knowledge crystallization* processes.

Furthermore, as described by Bertin [22] there are three types of readings or interpretation of a result and these are *Read Fact*, *Read Comparison* and *Read Pattern*. These types are just as they sound where read fact is about reading values e.g. out of a graph or table while read pattern is about looking at the overview. Read comparison is a mixture between read fact and read pattern as this is about the proportions and size between the data. What "reading" one uses depends on what kind of knowledge one wish to gain and will have an impact on all steps of the process, but in particular how to visualize the filtered data in the final step.

6 Design & Result

This section is about the actual process from ideas to implementation including changes that occurred during the process and special considerations that had to be accounted for. This part presumes that the reader is familiar with the technologies described in the previous chapter.

6.1 Initial Ideas

As part of the work I started by trying to create some own visual ideas by looking on the problem from what I thought could be useful. This included drawing some crude sketches of how to represent the data without much considering to the limitations of the system and time available. There were however some guidelines of the thoughts, and that were to allow for an overview of the entire fleet or large parts of the bus fleet. From there the user should be able to go into higher levels of details. Allowing the user to go from a big picture, down to smaller scenarios with fewer vehicles or even down into individual vehicles.

6.1.1 Map View

One of the early ideas involved tracing and summarizing how the vehicles were driven on the roads and where the fuel was wasted. I hoped to capture two things with this idea. The first being to locate idling such that if the bus had to stand still where there isn't a bus stop like red lights or a crossing. The second being about finding speed changes like acceleration and deceleration as this is a behavior that cost considerable amount of fuels and something one want to reduce. Reasons for this kind of behavior can be due to other traffic or a sharp turn in the route.

As for representing this, I thought of adding a layer on top of a map where a brush would color the roads according to different levels of fuel usage based on a summary for all vehicles selected, this could potentially be average or total fuel used. This could be a good way to visualize and assist the planning of routes and roads. It helps to find sections that cause unwanted halt in the flow of traffic and thus unnecessary emission and costs.

Figure 2: Simple illustration of how different color levels could illustrate the fuel usage for different sections. In this picture dark green indicates low consumption, green driving with color shifting towards red as the fuel usage increase. The image consists of a map from OpenStreetmap (© OpenStreetMap contributors, CC-BY-SA) with added fuel consumption from a fabricated scenario.

It should be noted that a harsh breaking and deceleration does not increase fuel consumption, however it does mean that one didn't plan for the route ahead as dictated by eco-driving. The reason is that after a deceleration, the driver usually wants to regain some its speed and this is done by acceleration which is fuel expensive. Therefore, harsh breaking will count towards high consumption as shown in Figure 2.

This concept could be expanded to cover other variables to plot in a similar way for example speed, acceleration and deceleration or engine speed (rpm) as this could be used to explain the pattern of fuel consumption. Another specific variable could be time spent in at position. This could however be seen as the inverse of speed but is probably easier to grasp when elaborating how the environment and obstacles affect the traffic.

Lastly allowing the display of altitude could be beneficial when trying to understand why a certain section is above average in fuel consumption as the altitude can easily be forgotten when looking on a two dimensional map. It should be noted that these variables are often co-related to one another and it could thus be quite useful to allow multiple variables to be rendered at the same time. But how to achieve this without cluttering the graph was not elaborated on.

An alternative way to visualize the data would be to filter it in a way that draws attention to certain critical and interesting regions instead of showing everything. One way of doing this would be with the use of thresholds that once exceeded would add a marker to the map. The marker would indicate that the region have for example a high fuel consumption. In order to avoid having too many markers in a region and thus confusing the user it would be preferable if the markers would be grouped into one (depending on the current zoom level). This would allow for an overview with more details when zooming into an area. This is illustrated in Figure 3 and Figure 4.

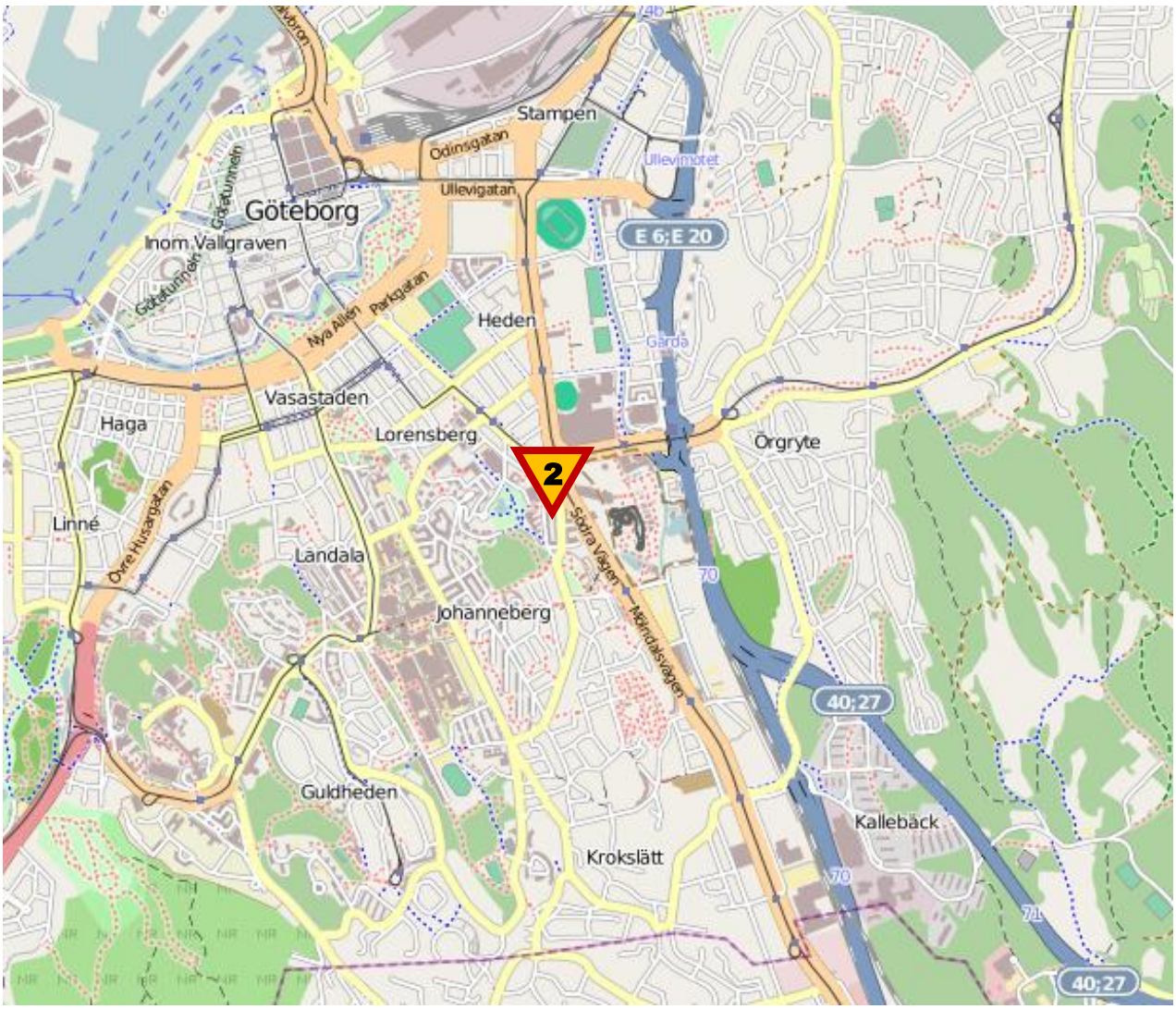


Figure 3: An overview image illustrating when zoomed out. It shows two sections in the Korsvägen region with high fuel consumption. The image consists of a map from OpenStreetMap (© OpenStreetMap contributors, CC-BY-SA) with markers added from a fabricated scenario.

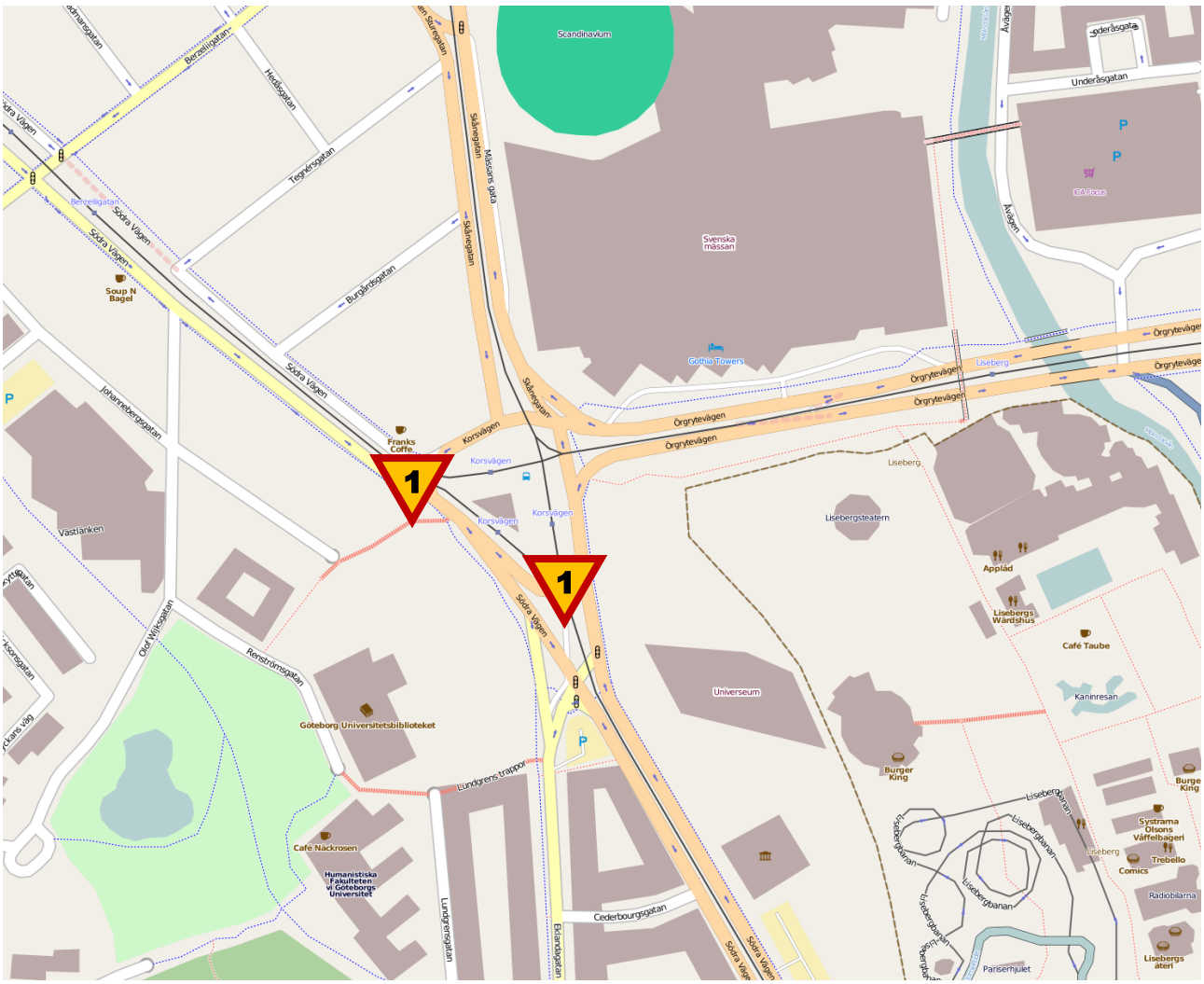


Figure 4: Illustrating details when zoomed in. The markers have divided to provide more details as to satisfy the zooming level. The image consists of a map from OpenStreetMap (© OpenStreetMap contributors, CC-BY-SA) with markers added from a fabricated scenario.

This second approach is probably better suited for the real world considering measurement errors and precision in GPS and odometer. Another benefit is that it would require less data to give a good estimate.

6.1.2 Chart View

Inspired by the functionality of the existing system I thought of adding a plot that based on the same search criteria as the map could show the data with its numerical value. This could then supplement and be used in conjunction with the map. This inspiration came from two functionalities where the first being able to plot the data from one vehicle with respect to the time. And the second functionality was being able to draw the vehicles positions in relation to their current route.

The idea was to plot a summary over more than one vehicle in a chart instead of on the map. To be able to do this in a useful way the data needs to be grouped along the x-axis. One possibility was to let the x-axis represent the logical position for a given route then this would allow the data to be grouped in one dimension (a map having two dimensions). And this should also correspond to position on the map and thus enables one to go between the map view and chart view. Such mapping between the views could however be problematic as just having a logical route position might not always correspond to where the vehicles really were, both due to measurement errors and deviation from the route. It could be reduced by filtering or using several sources of information.

Additionally one could also mark where the actual stops are along the x-axis which would give a better understanding of the data. While on the y-axis, one could want to plot several values at the same time as they are often co-related. But unlike the map, this is usually easier in a chart as one can have several y-axes and several series.

As for the representation of the data along the y-axis, there are many possibilities of how to visualize it. Among these are showing the raw data points in a scatter series, having line or higher degree of spline to represent it.

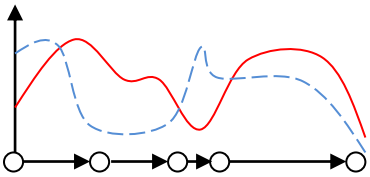


Figure 6: A spline plot with two parameters drawn. Exactly how the y-axis displays the value is omitted e.g. one could have additional axis on the right side.

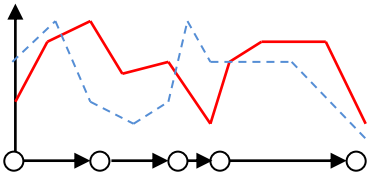


Figure 5: Line plot.

Considering one probably wants different types of summaries of the data to be shown a better choice could be to use some variant of box plots[23]. Another good reason for using box plots is that it shows the distribution of the data points and gives a better view of the underlying data. One common type of box plot is the five-number summary that shows:

1. Upper extreme
2. Upper quartile (75th percentile)
3. Median (50th percentile)
4. Lower quartile (25th percentile)
5. Lower extreme

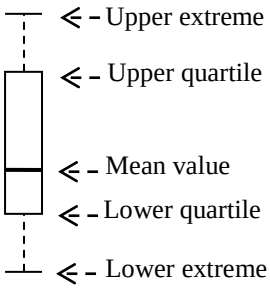


Figure 7: Illustrate the basic box plot. There are more advanced versions built from this one.

The upper extreme (1) and lower extreme (5) varies for different box plots but is normally the maximum and minimum, which would be fine in this case. As for the percentile, it means that that many percent of the values are below it such that 25th percentile means 25% of all the values are lower than this value and of course, the 50th percentile is thus the same as the median value.

As shown in [23] there are many variations of box plots with different levels of information but one should also be aware that at some point to much information can instead confuse or remove focus from the most important parts. This is especially true for the inexperienced users that can easily misinterpret such graphs.

Additional to these traditional ways to plot and represent data I had a wild idea of showing the distribution of data and density of the samples as a temperature map. What I mean by this is to render the data series in a similar fashion as they display the weather with clouds of different colors to indicate the heat or amount of rain for different regions. This could be useful in the case that the data is not normally distributed as to show different “paths” the data could be centralized around.

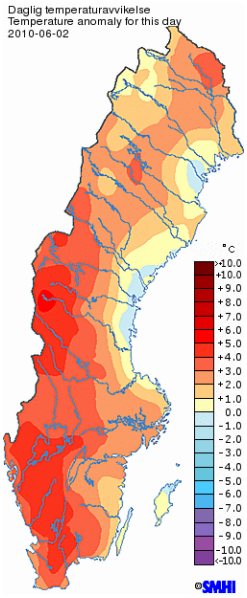


Figure 8: Example of a temperature map from SMHI[24]

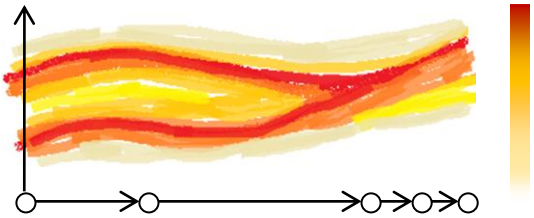


Figure 9: Red indicates a high density of data points and it reduces in color as the density reduces.

One reason why behavior could occur is in the case of a red light where some vehicles have to stop while others can pass unstopped.

6.1.3 Comparison View

Up until now, the views have been about tracing and understanding how the fuel is used when it comes to the environment and drivers. But it would also be desirable to have a view that supports comparing entities as to find changes that can improve the usage. Using this as a starting point, I can see two approaches or views that are based on this. Where the first would be a standard view where the user defines what to compare and gets a result. And the second approach is where the system automatically does certain comparison and then presents a set of improvements to the user. Of course, both of these are desirable.

The next important question is what should be comparable. The most obvious entities that a user would like to compare are the vehicles and the drivers, as both have been proven to have a large impact on fuel consumption in the pre-study, and they can be affected by the operator. As both of these categories can contain a large number of unique entities for a larger operator, it could be troublesome to distinguish a single entity from hundreds of others. A solution to this could be to group the entities on things they have in common in order to reduce it to a few groups. This means sticking with the previous design of getting an overview and then more details if needed.

Additionally it might be interesting to see how the fuel consumption is affected by the journey[1], that is narrowing the search certain to a specific route and time interval. Despite the chapter, 4.1

[1] A journey is associated with a route and time.

Parameters stating that the time on the day had little effect on the fuel consumption due to having dedicated bus-only lanes.
Furthermore leaving out the time in the comparison might be unfair since the environment changes such as number of passengers etc.

6.2 Implementation

The application design follows the n-tier architecture as described in the analysis where one separates the presentation (client application) from the logic and data. The latter two are made into a web service that makes its requests to a database and do some processing before returning the data.

6.2.1 Database

When creating new tables in one of the databases one needs to consider the vast amount of data and that the databases are distributed. Due to these conditions, there is a need to keep the queries simple and reduce the amount of data to maintain a low response time. The consequence is that one often wants to pre-process the data and store it in as easy and accessible format possible inside the table such that it matches what the user and client will query. To achieve this one solution is to have one table for each report or summary of the application. The disadvantage is having redundant data but the advantage is low response time. As a result I pre-calculated some of the values to allow data to be joined within reasonable time; this was the result of running the application and experiencing unreasonable delays.

As for collecting data into the tables and generating the reports, I started out by creating a separate program that would gather data from the other tables and then do some processing and filtering of it. The resulting reports could thus be seen as a snapshot of the world from when the program ran. This was later on changed into using the framework that Consat had constructed for keeping the tables updated with the data reported from the vehicles. It works by having the vehicles record data that is not needed in real-time into log files. These log files is at the end of the day uploaded to the Depot Server where they are parsed by the framework. This allows the program to insert and update entries that are the base of the reports used by the application.

When doing this parsing I encountered problems several times and found that real-world data is often not as expected. An example of such an issue is where the orders of the stops were wrong. Where my interpretation of the recorded events was that the drivers missed several stops but realized this later on and and went back to correct the mistake. Another issue was that the drivers could change schedule and journey in the middle of a trip and later changing back. An example would be performing journey A, then completing journey B and finally resume journey A again. These are cases that if not considered could give an incorrect view in the application. Thus, it is important to try and guard the application from as many of these as possible. The simplest solution to guard against this is to cut the route when it is no longer being followed as predicted. This strategy allows the application to make the most use of the data while still keeping it simple and throwing what it considers invalid data.

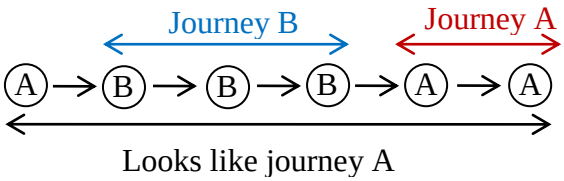


Figure 10: If unchecked one might accidently include B into the calculation of journey A.

To access the database there are several different methods as described in the analysis. Among these methods, the one found to be most suitable was Linq which is a standard API and library in C# for persistency and is described in chapter 5.1. It gets the location of the database to use by defining a connection string. This connection string is then used as a key to lookup an entry defining the actual connection. The definition is user defined and located in an external XML configuration file. The reason for using Linq as DAO is the simplicity to get started and having access to everything very quickly. The disadvantages becomes greater if there are a lot of changes in the program life cycle due to the tight coupling between the code and the database, but that was not a major concern for this particular project.

6.2.2 Web Service

Creating a web service in Silverlight is quite simple as there is a great support by the framework. When creating the web service the logic is just as any other code one writes with some metadata added to the classes' one wish to return. In accordance to the principal of SOA the transportation of the messages and other environment dependencies are put into and external XML document where one can configure these even after compilation. This includes connection strings for the database as described in previous chapter. As for the type of web service, it became more of an RPC directed service despite not having a specific state saved about the client at this moment of time. But this allows later changes to include username and password as for accessing.

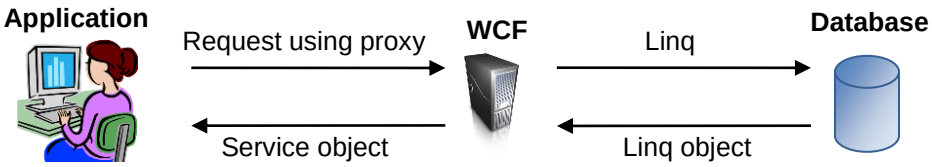
The configuration of transportation protocols does however affect the service even in design as there are limitations of the transportations such as the amount of data that can be transferred. When encountering such a barrier there are a couple of options to overcome this such as:

- Look for a different protocol that allows more data to be transferred.
- Make selection of data entities a part of the contract and use several requests to receive more data when needed.
- Try to compress the data returned or reduce the amount of data required at each point in time for the views.

Another design issue with regard to the n-tier architecture is where different parts of the logic should be placed. Normally one would want to put as much logic at the web service as this reduces the bandwidth needed and to get a portable application where code can be reused. The disadvantage of that is a potential bottleneck due to heavy computation as discussed earlier, but also a less generic service. Furthermore, by processing the data a lot on the web service, the service becomes less general and thus one need to have more services if one want to process it differently.

The choice I made from these two previous considerations was to have the web service functionality layered without any processing at the bottom layer and more processing as one moved up the layers. Making the higher layers returning the processed data and thus lowering the resulting data size and avoids the transport limitation. The disadvantage is that the application needs to make a new request whenever the user changes view. The result is more requests but less data for each request. Furthermore, as to keep a good design and allow for changes, there are different objects for each of the tier. The WCF uses a Linq objects as "contract" to the database and another set of object

as the contractual object provided by the service to the application. The application in its turn transforms the service objects into local application objects.



6.2.3 Summarize Data & Route Position

In section 6.1, when explaining different views and ideas it was not said how the grouping of data in the graphs would work other than using a logical route position. Precisely what it meant and how to calculate it was not defined. It did however say it is an indication of progress made on a route. But it was not said how to measure this progress as it was left as an implementation decision. Looking at the available variables this could be done with one or a combination of variables e.g. time since departure, distance traveled and GPS coordinates. The choice fell to making logical route position equal to the distance traveled since the start but truncated and bound. It is truncated so that it cannot pass a stop no matter the distance traveled, unless it has actually physical passed (as reported by the vehicle). Thus, the logical route position is bound by the length of the link between two stops at all time. The reason why using the traveled time was found inappropriate is that it can take different amount of time to travel a given route even though the distance is the same. Another option would be to use the GPS coordinates; however, one needs to consider the measurement errors that GPS inherently have. Furthermore, it could be tricky to convert GPS coordinates into some form of route progress. One such scenario could be when a route has two roads in separate direction and in close range to each. This scenario could force the use of additional parameters e.g. a direction parameter to complement the GPS in determining progress.

6.2.4 Views

The final integrated application consists of a searching page where the user defines a set of criteria for data selection. The search criteria are connected or transferred to a comparison page in the second tab and driving page in the third tab. The comparison page is implements drill down chart for getting an overview of how the data fuel consumption is distributed between groups and vehicles. The driving tab on its hand show how the data correlates to positions on a map. By having the data shown on a map one identify hot spots, areas that have a tendency to consume more fuel if passed by.

6.2.4.1 Search View

Using a left to right flow of interaction the user can select the most important search criteria to be used for the views in the application. This is later supplemented with individual drilldown on the pages to further narrowing the search. The page layout and workflow is centered on being able to quickly select criteria. As part of this principle, certain parts of the page have a default selection to more quickly navigate forward. Another aspect as discussed in earlier chapters was to group vehicles and drivers as there can be a vast number of these, and groups would reduce the number of entities shown simultaneous. An example of this is the hierarchical design of the vehicle selection where one can find a vehicle in several groups and they are grouped by vehicle model (type) as well as a logical grouping done by the operator, in another application. A similar structure should be

done for the drivers but this was not done during the project.

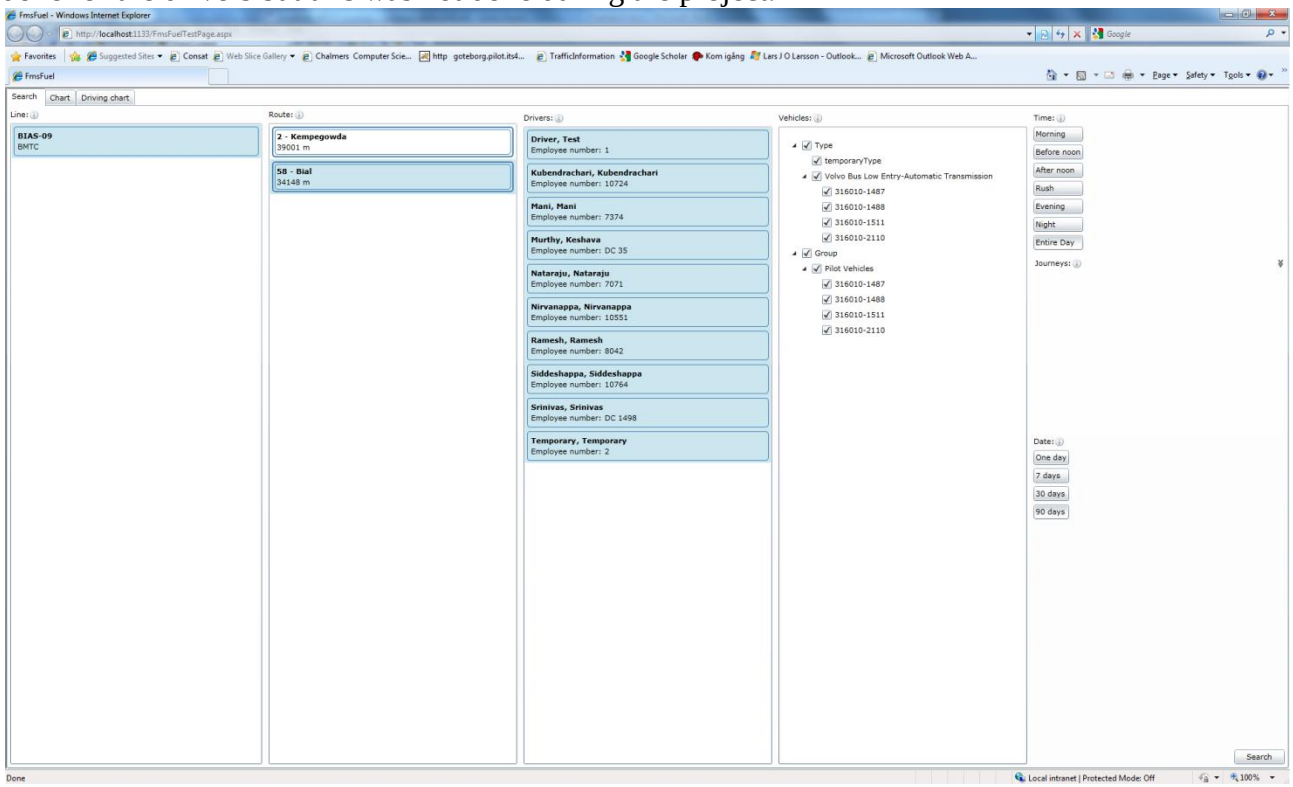


Figure 11: An overview picture for defining searching criteria with a flow of selection from left to right and default selections to allow for quickly searching.

An example of the architecture and workflow is that it requests data as it needs it. Looking on the first column from the left as in Figure 12 there is no line selected and thus no routes displayed in the second column. Once a line is selected, it fetches all routes associated with that line as illustrated in Figure 13.

Figure 12: Search view with no lines selected and thus no routes available.

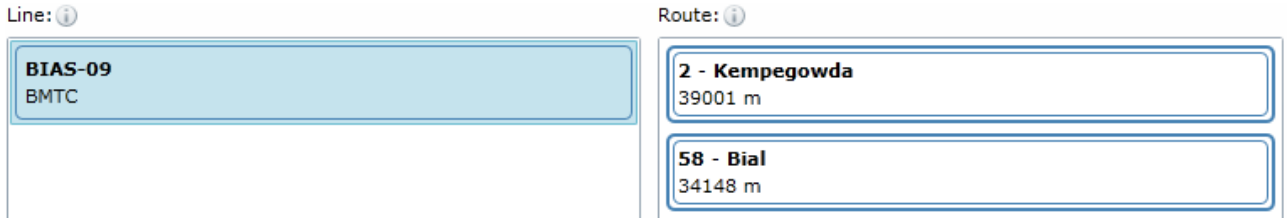


Figure 13: Search view with a line selected and a result has been received with available routes for that line.

6.2.4.2 Comparison View

As for the view when comparing the entities, the layout is divided into three parts, as shown in Figure 14.

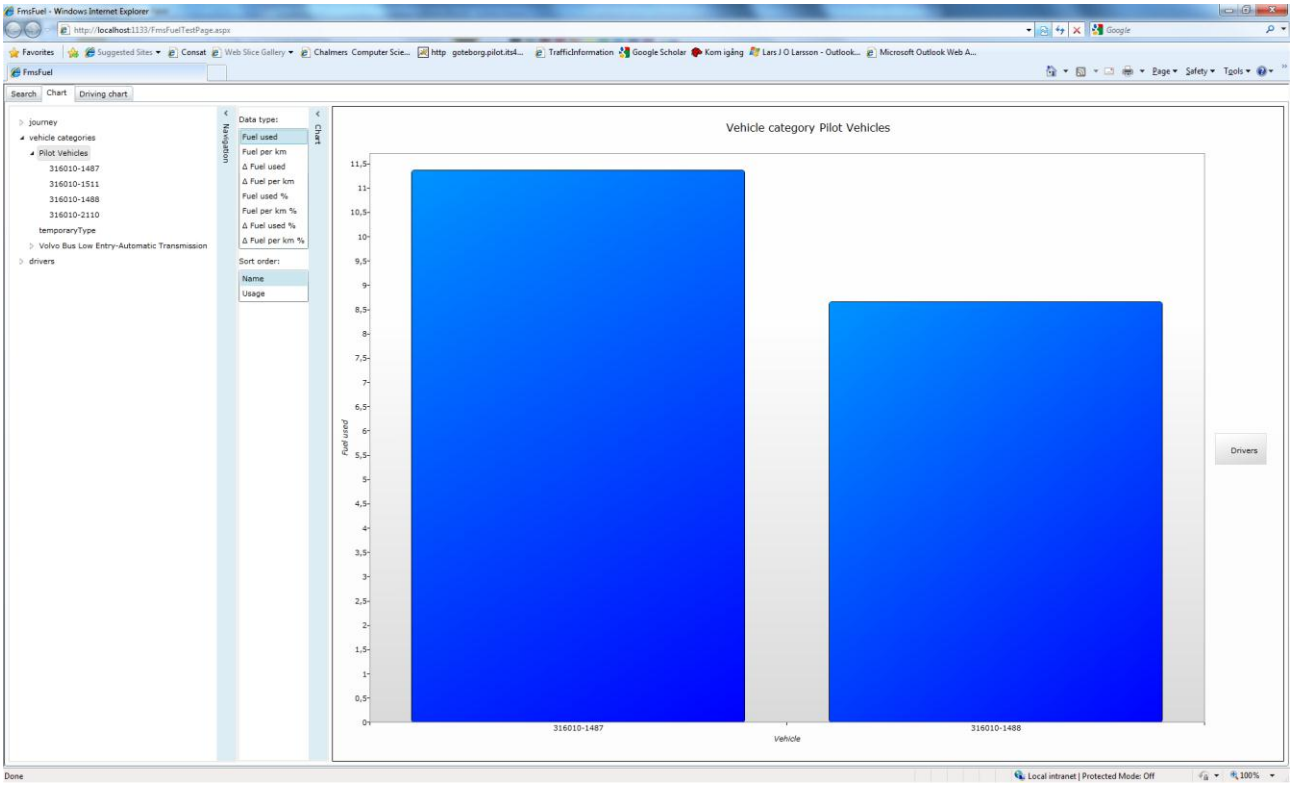


Figure 14: Comparing of different data such as demonstrated here there are four vehicles in a common group that are being compared. However, two of the vehicles lacks data for the given search parameters.

Starting from the left is a navigation tree used to set what type of entities to compare and at what detail level. This is not the only way to navigate as one can also navigate downward the tree by clicking inside the chart on the column that one wishes to expand and see the details for, this is known as drill down. This design allows for a high overview at the three outmost nodes, journey, vehicle categories and drivers, with more details as one go down. In the leaf nodes (the deepest nodes), one can only see the historical changes for that specific entity. That is how the fuel used for this entity has changed over time as shown in Figure 25.

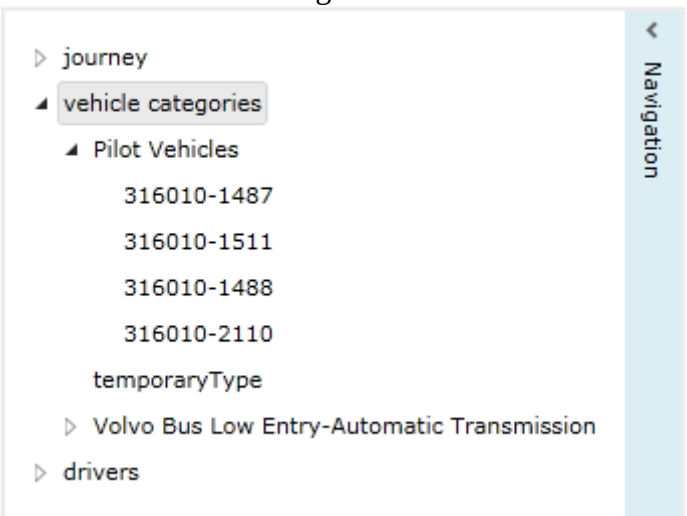


Figure 15: Shows how the navigation tree for comparison view looks like. The tree is based on the three outmost nodes that symbolize the three categories of entities.

Next to the navigation tree comes a data selection tool that differs slightly for the nodes in the navigation tree. The data selections are based on the total fuel used and fuel per km. It allows slightly different views of this data and thereby highlighting different aspects such as how many percentages more or less that was used for the entities compared to that of the average.

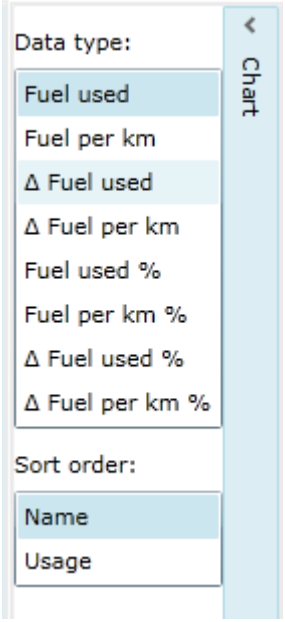


Figure 16: Of the data selection tool, that affects the rendering of collected data. This tool is slightly different depending on the location in the navigation tree.

The last part of the view is the charting view that displays the data from the choices made. This is also dependent on current navigated page as to how to display the data. When comparing entities or groups it is done as a column chart with the names of the entities on the bottom x-axis and the selected data on the left y-axis. When moving the mouse over a column a describing tooltip will appear with a brief summary of the column as shown in Figure 17.

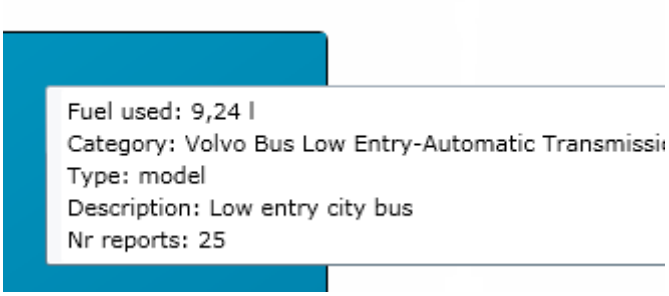


Figure 17: Illustrate the information popup for a column when the mouse hovers above it. This picture is for vehicle category in the navigation menu.

The menu is shown as tree where the root node is the least detailed and gives an overview as illustrated in Figure 18 where you can see data for categorizes of vehicles. The leaf nodes of the menu are the most detailed showing data for only a single vehicle. A category can be either a

specific bus type or a logical grouping of busses that is defined by the user or system.

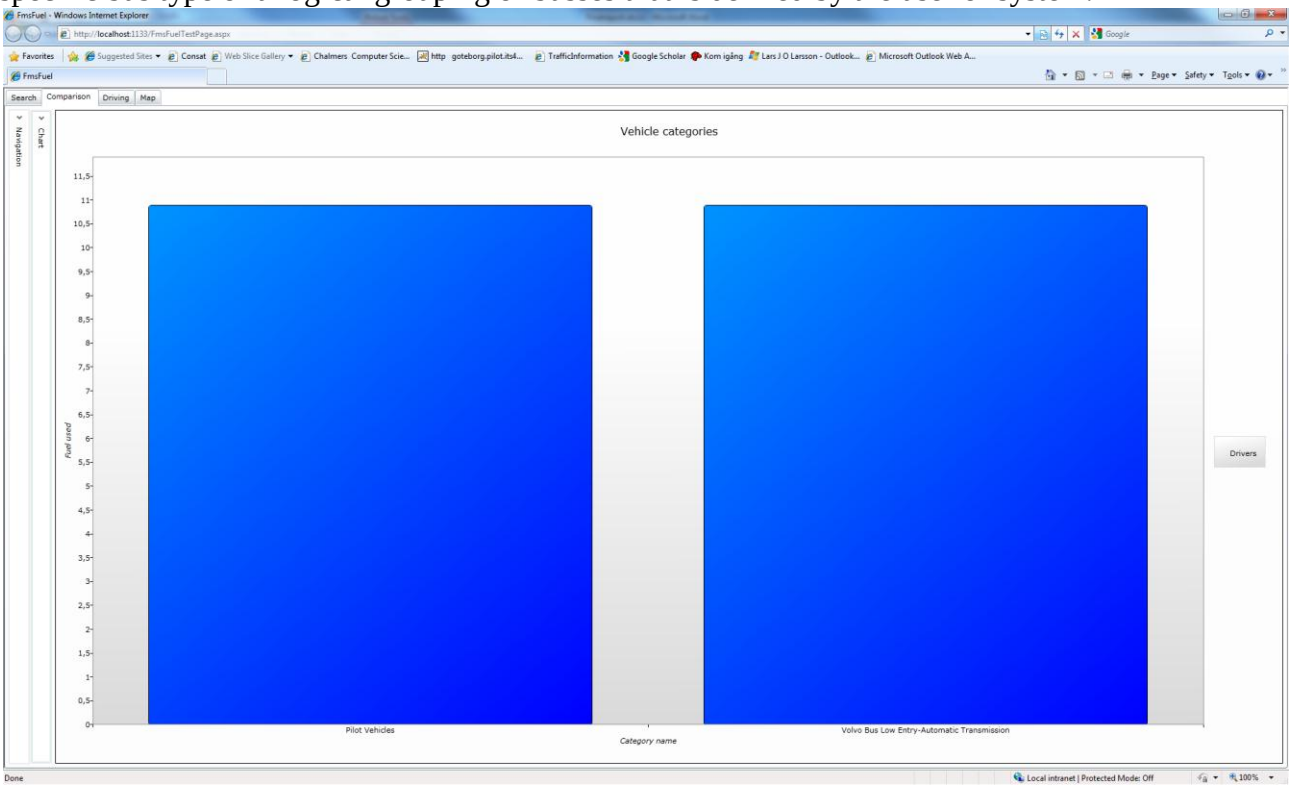


Figure 18: An overview of the vehicles by summarizing the data on categories of busses.

The next level is slightly more detailed due to further filtering of data and displays only the the vehicles for one selected category as demonstrated in Figure 19.

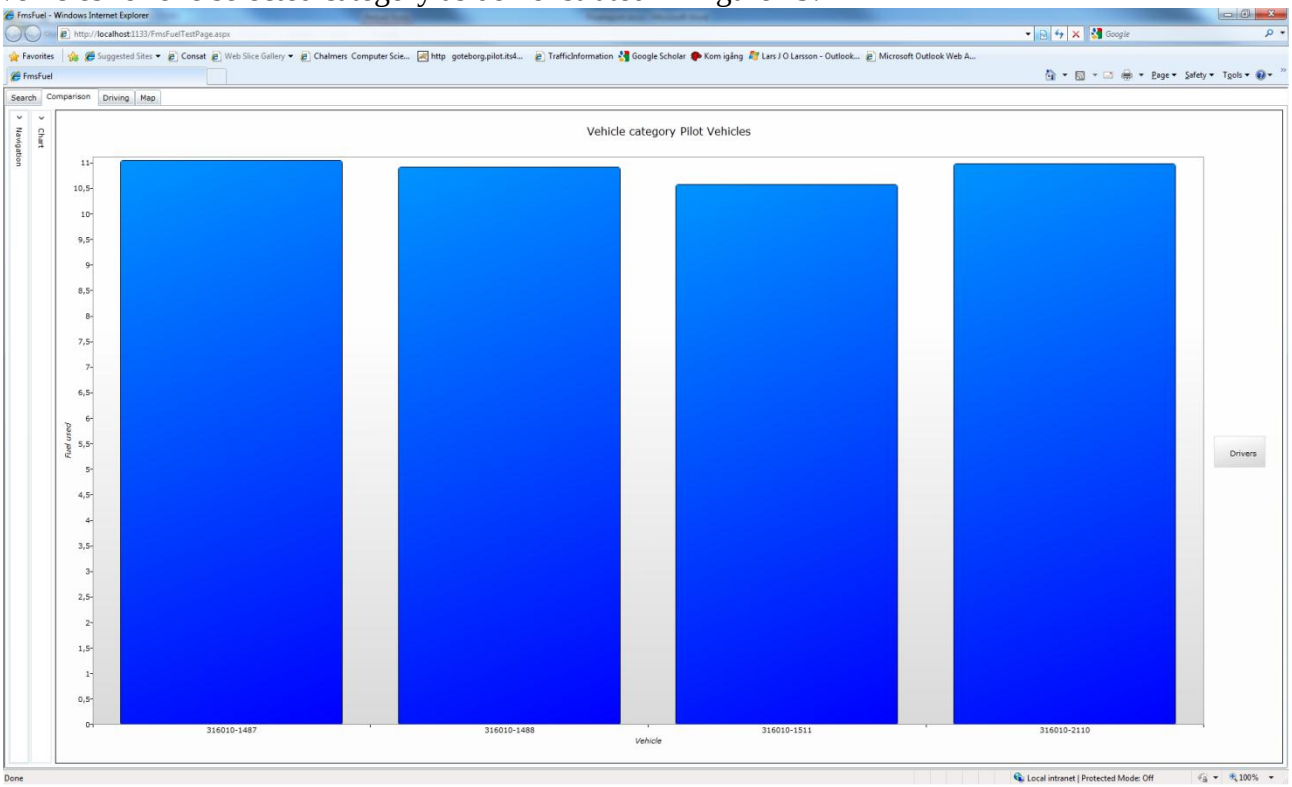


Figure 19: Vehicles for a given category that can be either a bus type or a group of busses.

Finally, at the most detailed level there is a history graph for a single vehicle as shown in Figure 20.

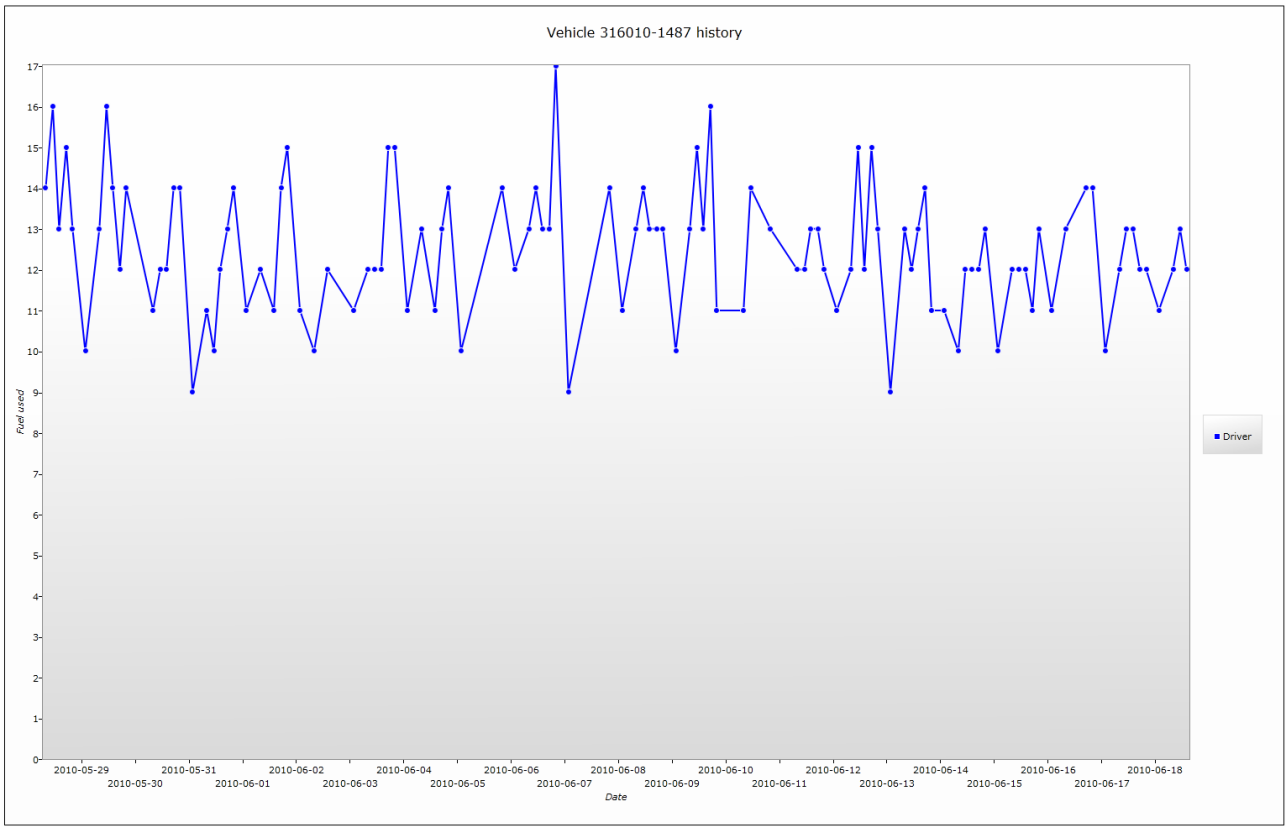


Figure 20: History graph for one vehicle.

Furthermore, the effect from the selection of sort order is demonstrated using the entity type of journeys. First having the sort order based on the time of the day, the data is resorted and the value displayed along the axis is changed to start time as illustrated in Figure 21. The second example is where the columns are sorting according to fuel usage as shown in Figure 22.

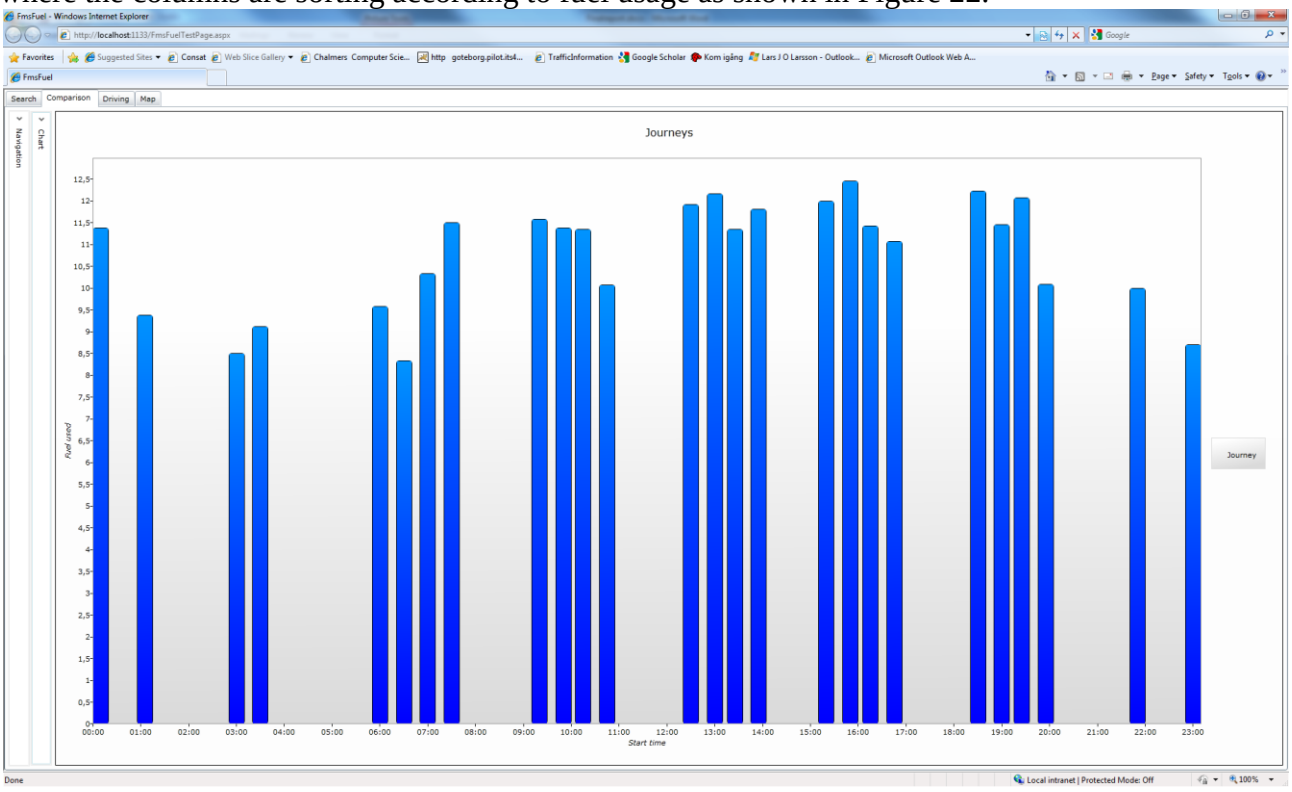


Figure 21: The data is sorted according to start time and the x-axis is thus the time on the day.

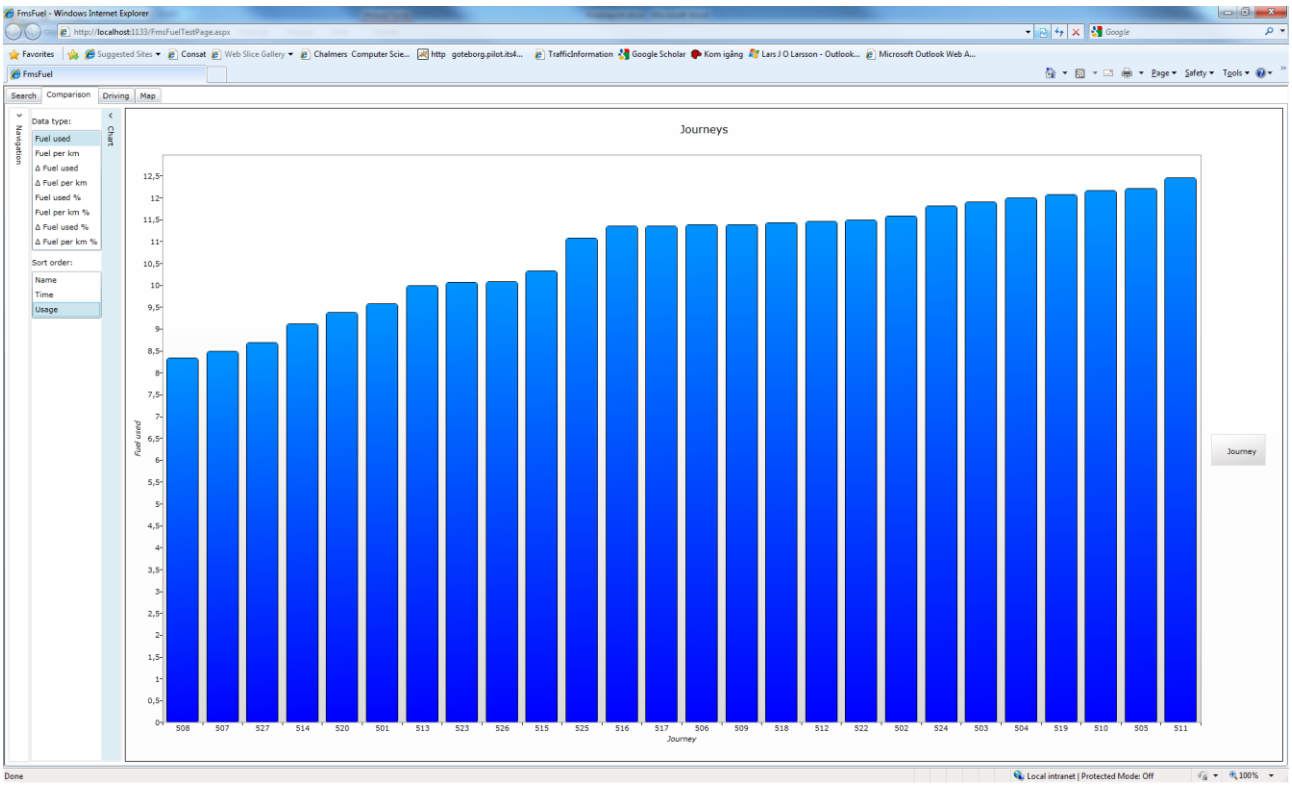


Figure 22: Using sort order based on fuel used and displaying the name of the journey which in this case is a number. The journeys are for the same route and same path.

Finally, the data selection tool is used to highlight certain aspects of the data by manipulating and transforming it slightly to improve the visualization of the search criteria. An example of this is the ability to show the *delta fuel* used for the entities. The delta fuel is calculated by comparing it to the average of all data that matches the search criteria and thus enhances the presentation when doing a "read comparison" as mentioned in chapter 5.5 and the result is illustrated in Figure 23.

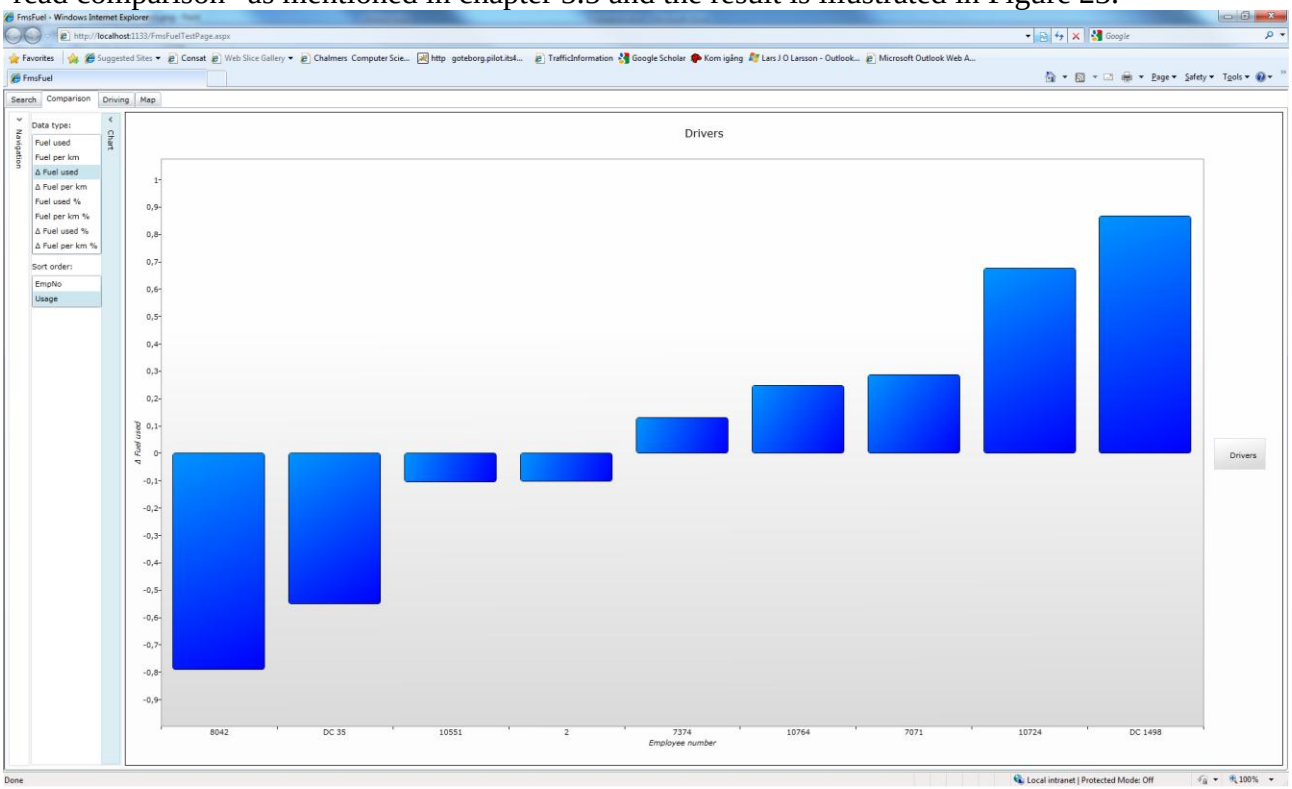


Figure 23: This graph is for driver entities and displays the delta fuel used compared to average for the route and given search parameters.

As for the drawing of the charts, I started out using a library called *Visifire* [25]. It was simple to use and easy to get going however it suffered from certain limitations. Furthermore, there were a few bugs and issues as the support for Silverlight 4 for Visifire were still being developed. Visifire was therefore replaced by *Silverlight Toolkit* [26] as the charting library. This was due to Silverlight Toolkit allowed a greater flexibility and customization of the charts such as styling and tooltips. Silverlight toolkit and Silverlight 4 was still in preview however, they were quite stable even though they both had a few issues. Silverlight toolkit is developed by Microsoft as part of their open-source effort and it is free of charge both for commercial and none commercial use.

7 Discussion

The result is a working prototype of an application with one search frame and one comparison view of the fuel consumption supplemented with a crude driving view where the data is bound to a position on the map. The objective was to create an application that provides different levels of details concerning fuel consumption of bus fleet. The goal was that this would allow the bus operators to lower their driving costs something that unfortunately hasn't been evaluated at production environment. The final application allows easy access of fuel data and drilling down into the details from overview perspectives. The user can drill down using either through the menu system or by clicking directly inside the charts. But it is by no mean a finished product even though some styling and customization has been done. Examples of styling and customizations are the drawing of data series such as columns and lines as illustrated in Figure 25. Another customization is the use of individual tooltips for each of the entity types in the comparison view. The tooltips are customized to display information corresponding to the entity type and detail level. The workflow is also polished both to be user-friendly and allowing experienced users to quickly get their work done. In order to improve speed and the work process for experienced users certain considerations has been taken including: avoiding window pollution, avoiding pop-ups, and preferring list boxes over combo boxes.

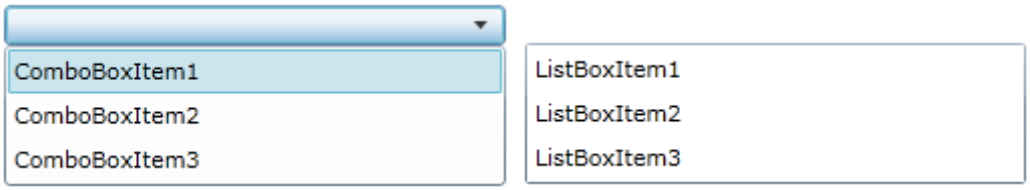


Figure 24: Illustrates the difference between a combo box and a list box.

As for the choice of language, I found Silverlight it to be very clean and neat. However, it should be noted that Silverlight despite being version 4 still lacks certain basic and useful features that WPF has. I do however suspect that the platform will continue to grow and evolve with more of the functionality that can be found in WPF and thus bridge this gap between browser and native platforms. The choice to use Silverlight can overall still be considered a good choice even though it would probably have been easier to trade the browser compatibility of Silverlight with the extra functionality of WPF. But since I haven't worked explicitly with WPF, it is hard to tell with certainty. I do believe that Silverlight will be an important platform in the coming years both due to its maturing process and the fact that Microsoft is pushing for Azure, Microsoft's cloud platform.

7.1 Data Behavior

The task became more difficult than I had initially thought as the data suffers from unpredictability. In a few cases the drivers seems to behave irrational and unexpected. This can cause a headache in avoiding this from having a negative impact on the application. I spent a lot of time tracing the data and trying to filter or resolve these issues and while it worked out quite well it could of course be improved. Looking back, I can see the use of another more detailed level as to allow a quick explanation of why one tour or journey used ~17 liter of fuel while the next one only used ~11 liters of fuel as shown in Figure 25. This could probably be explained by the other views or a new and more specific search, but having a quick access to the underlying data would be handy.

Another issue is that the sampled data have a tendency to change values rapidly and thus if plotted it can look very spiky and thus make it harder to interpret as illustrated in Figure 25. This behavior is unavoidable as the busses are often used in city traffic where there are short distances between stops and thus forcing the driver to acceleration and deceleration which in turn gives rapid changes to the sampled data values. It could in some aspect be hidden in the presentation by using either a filter or transformation on the data in order to give it a smoother transition. Another option is to use a higher degree of spline function when plotting. This could then be adjusted to different zoom levels, so that a higher zoom would mean closer to the sampled data and a smoother curve for a

lower zoom level. Unfortunately, this has not been done and there remains spikiness in the application.

Figure 25: History of fuel usage for one vehicle. One can see changes in data and in this case getting a zero might first indicate an error in the application however, this is due to a bug in the FMS-bus of certain bus models and the broadcasting of incorrect values.

7.2 Comparing

For comparing entities one of the fundamental issues is to make sure the data being considered is compared under as equal conditions as possible to avoid giving meaningless or misguiding results. On the other hand, it is important not to limit or deform the data as the result could become distorted. That is to get the data to fit the model and criteria of the application view while at the same time avoiding making invalid transformation or having to throw a lot of data. One thing I learned is that very little of the real world data does fit precisely into ones models or expectations. A portion of deciding data equality is put on the user as it has to select what hours of the day, what drivers and vehicle for a given route that is of interest to compare as illustrated in Figure 11. While some other criteria are set by the program itself, as how much of given route that has to be completed to be used as a data source. This selection that the application does is an estimate from the logged data that the vehicle uploaded. There is no exact science or silver bullet for solving this. The application provides a result and it seems to be decent enough for most cases, however one should not stare blindly at the result but rather use it with other sources.

7.3 Limitations & Improvements

There are currently limitations both on the existing systems and the project application. There is of course a limitation on the resolution of the data measurements by the bus. But more importantly is that there is a limitation on the amount of data being recorded into the logs and used by the system, this is to make it manageable and practical. What this means is that not all the values broadcast by CAN-gateway are saved into the log files e.g. the odometer value is saved on important events or after certain configurable thresholds. Example of such a threshold could be that the odometer will be recorded if it is more than 100 meter since the last entry. These records are later used by a parser that inserts that data records to the database that is used by the application. These thresholds are set

individually for the variables and thus creating situationsz where several speed samples are mapped to the same odometer value and thus the same logical route position.

Furthermore, there are existing bugs on the data transmitted by the CAN-gateway for certain busses but this issue has been disregarded and can thus give invalid values or create confusion in the application. Furthermore, there are several existing versions of the BUS FMS-standard and unfortunately, the latest isn't fully agreed upon or implemented in all busses. The later version of the standard includes key variables such as *fuel rate*[2] and *instantaneous fuel economy*[3] but these could not be used. Finally, several important variables are unaccounted for such as weight, weather and conditions of the vehicle etc.

As seen in *design & implementation* there are suggestions for several views where only a few have been implemented. Some of the views need to be further analyzed before implementation.

There is also only support for English and no effort has been made to translate or prepare the program for other languages as stated in the limitations.

[2] From the standard, an indication not a contractual value [5] of *"Amount of fuel consumed by engine per unit of time"* and is a measurement of L/h.
[3] From the standard, an indication not a contractual value [5] of *"Current fuel economy at current vehicle velocity"* and is a measurement of km/L.

Bibliography

1. Euro Bus Expo Enviroment Brochure - Everyone needs to travel. *Euro Bus Expo.* [Online] [Cited: June 7, 2010.] http://www.eurobusxpo.com/client_images/Environment%20Zone%20Hand%20Book.pdf.
2. **ECODRIVEN project for Intelligent Energy Europe.** *ECODRIVEN Campaign Catalogue for European Ecodriving & Traffic Safety Campaigns.* 2008.
3. **WEILAND, RICHARD J. and PURSER, LARA BAUGHMAN.** *Intelligent Transportation Systems.* [Document] s.l. : Transportation Research Board, 2000. url: http://pubsindex.trb.org/view.aspx?id=639268.
4. **Consat Telematics.** ITS4MOBILITY - Technical Description. 1.0 December 2009.
5. *Bus FMS-Standard.* s.l. : Working Group Bus FMS-Standard, 2007.
6. **Alldritt, Noel.** ITS4mobility Fleet Management. 04 - version 1.0 - 2007-01-25.
7. Kollektivtrafik. *Hogia.* [Online] Hogia. [Cited: June 8, 2010.] www.hogia.se.
8. *A STUDY ON THE FUEL-CONSUMPTION CHARACTERISTICS OF PUBLIC BUSES.* **FWA, B. W. ANG AND T. F.** 12, April 18, 1989, Energy, Vol. 14, pp. 797-803.
9. **Sandberg, Tony.** *Heavy Truck Modeling for Fuel.* Electrical Engineering, Linköping University. 2001. Thesis No. 924. ISBN 91-7373-244-3 / ISSN 0280-7971.
10. *Training urban bus drivers to promote smart driving: A note on a Greek eco-driving pilot program.* **Zarkadoula, Maria, Zoidis, Grigoris and Tritopoulou, Efthymia.** 6, 2007, Transportation Research Part D: Transport and Environment, Vol. 12, pp. 449–451. ISSN 1361-9209.
11. **Ericsson, Kerstin.** Hjälpmedel i bilar sparar pengar i Skåne. January 2010, 1, p. 28.
12. ISA – Intelligent stöd för anpassning av hastighet. *Vägverket.* [Online] [Cited: June 7, 2010.] http://www.vv.se/Startsida-foretag/Trafiken/Hastighet/ISA/.
13. **Reese, George.** Distributed Application Architecture. *Database Programming with JDBC and Java, Second Edition.* Second Edition. s.l. : O'Reilly & Associates, 2000, 7.
14. Web Services Architecture. *W3C.* [Online] World Wide Web Consortium (W3C), February 11, 2004. [Cited: June 07, 2010.] http://www.w3.org/TR/ws-arch/#relwwwrest.
15. **Juval, Löwy.** Programming WCF services. *Programming WCF services.* 2nd Edition. s.l. : O'Reilly Media, 2008.
16. Java BluePrints - Model-View-Controller. *Sun Microsystem - Java.* [Online] Oracle (Sun Microsystems). [Cited: June 7, 2010.] http://java.sun.com/blueprints/patterns/MVC-detailed.html.
17. **Reenskaug, Trygve.** MVC - XEROX PARC 1978-79. *Trygve M. H. Reenskaug .* [Online] University of Oslo. [Cited: June 7, 2010.] http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html.
18. **Boodhoo, Jean-Paul.** Model View Presenter. *MSDN Magazine.* August 2006. URL: http://msdn.microsoft.com/en-us/magazine/cc188690.aspx.
19. **Martin, Fowler.** Presentation Model. *MartinFowler.com.* [Online] [Cited: June 7, 2010.] http://martinfowler.com/eaaDev/PresentationModel.html.
20. **Gossman, John.** Introduction to Model/View/ViewModel pattern for building WPF apps. [Online] [Cited: June 7, 2010.] http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx.
21. **MacDonald, Matthew.** *Pro Silverlight 3 in C#.* s.l. : Apress, Apress. ISBN 1430223812.
22. **Card, Stuart K., Mackinlay, Jock D. and Ben, Shneiderman.** *Readings in information visualization: using vision to think.* s.l. : Morgan Kaufmann, 1999. ISBN 1558605339.
23. **McGill, Robert, Tukey, John W. and Larsen, Wayne A.** Variation of Box Plots. *The American Statistician.* February 1978, Vol. 32, 1, pp. 12-16. url:http://links.jstor.org/sici?sici=0003-1305%28197802%2932%3A1%3C12%3AVOBP%3E2.0.CO%3B2-9 ; free source: http://lis.epfl.ch/~markus/References/McGill78.pdf.
24. Daglig temperaturavvikelse, Juni 2010 . *SMHI.* [Online] Sveriges meterologiska & hydrologiska institute, June 2010. [Cited: June 7, 2010.] http://www.smhi.se/.
25. **Webyog Inc.** Visifire. *Silverlight, WPF & WP7 Chart tool.* [Online] http://visifire.com.

26. **Microsoft.** Silverlight toolkit. *CodePlex.* [Online] http://silverlight.codeplex.com/.