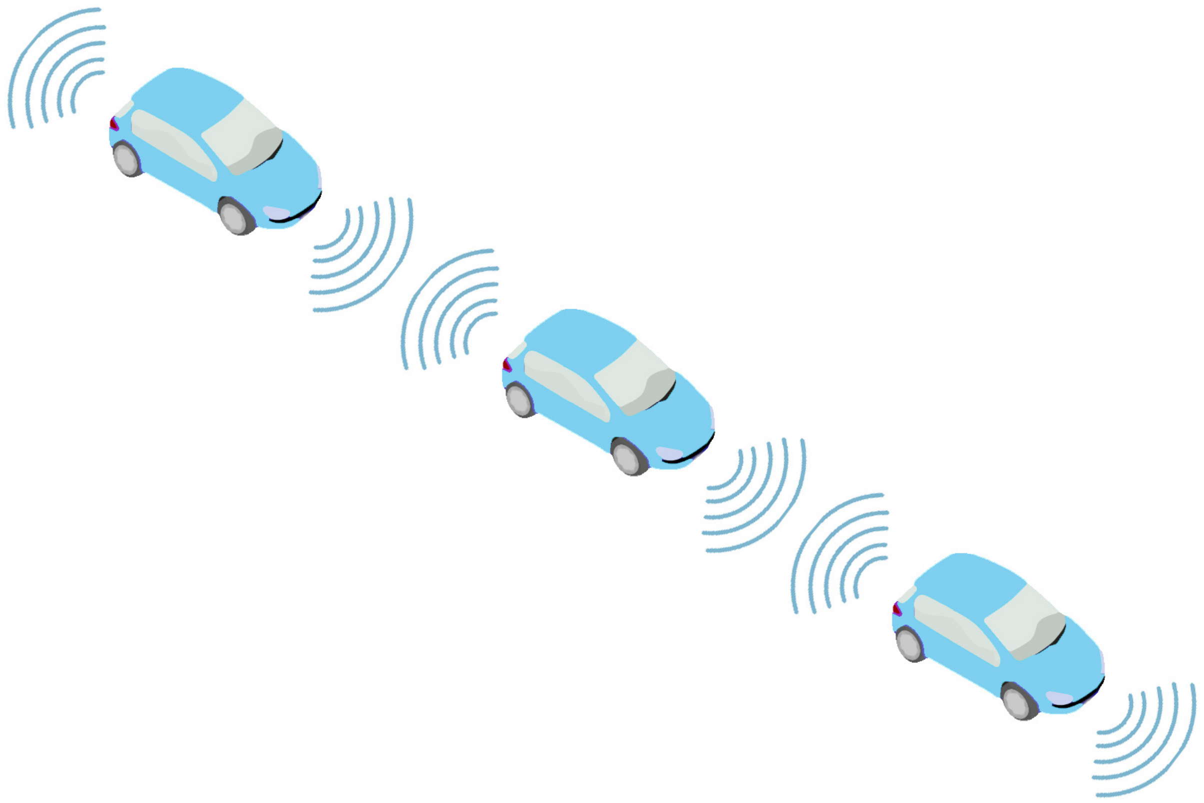
Talking Motion

Demonstrating communication in a vehicular environment



Ruud van Heugten
Meijel, 10 June 2009

Graduation report for Fontys University of Applied Sciences

Student information:

Name:	R.J.G. van Heugten
Student number:	2056553
Study:	Computer Science and Engineering (Fulltime)
Graduation period:	2 February 2009 till 3 July 2009

Company information:

Name:	Fourtress BV
Department:	Software Engineering
Location:	Eindhoven
Supervisor:	N. Vos
Technical support:	B. Peeters

Tutor information:

School:	Fontys University of Applied Sciences
Location:	Eindhoven
Name:	G. Dirks

Paper information:

Title:	Talking Motion, demonstrating communication in a vehicular environment.
Date:	10 June 2009
Confidential:	No

Approval by company supervisor:

Name: Date: Signature:

Preface

This report is written during my graduation period at Fourtress BV for the Fontys University of Applied Sciences. It describes the process of getting started with the assignment and what I have done, up to finishing it at the end of my graduation period.

The assignment I have done is part of the Connect & Drive project which utilizes wireless communication between cars to improve safety and flow through on the road and decrease the length of traffic jams. To accomplish this, eight parties work together to create a cooperative adaptive cruise control system in which Fourtress is responsible for the software application. My assignment was to create software that could demonstrate the use of the project and also could be used in a further stage during the Connect & Drive project.

I would like to thank Fourtress for giving me the opportunity to graduate at their company. I would also like to thank all the employees of Fourtress who helped me make my graduation as successful as it is. In particular Niels Vos as my supervisor and Bruno Peeters for his technical assistance during my graduation period. Without their help it would have been a lot more difficult to get everything working as it is now. Also with the help and support of Geert Dirks as my tutor for Fontys Hogescholen I can graduate as Bachelor of Information, Communication and Technology.

Meijel, June 2009

Ruud van Heugten

Table of contents

Summary

Nowadays there are a lot of traffic jams which grow gradually in length over time. Possible solutions are increasing road capacity or spreading the traffic more over the day. Although these solutions provide some decrease in traffic, both solutions have flaws. Another possibility is using the road system better and improve the flow through of traffic. To investigate such a solution Fourtress BV is working together with a consortium of other companies and technical universities on the Connect & Drive project.

The Connect & Drive project wants to utilize car to car communication to create a cooperative adaptive cruise control system. This system makes it possible for cars to use the communication to calculate the optimal driving speed and warn others for possible hazards ahead.

An important part of the project is the network communication between cars. To start testing and look into possible problems that can occur during the communication Fourtress wants to have a system that simulates final system. These systems must be able to inform each other with information like the current speed, GPS position and direction without user interaction. Since hardware is not available at the start, the application uses modules that stub this hardware and can be swapped later on.

At the start of the assignment a plan of approach is made to have a guideline during the project. At the start the main goal is research about what is necessary for car to car communication. The next step is to define modules for more dynamic and communication without user interaction. When this is possible, modules for simulating a real car can be implemented.

During the graduation project the main point of interest, making dynamic network communication possible, was pursued. This was done by utilizing a network library to send and receive network packets. With these packets it was possible to determine where other instances of the software application were and after discovering them, communication could be made. When communication was possible and everything worked ad-hoc, the other modules needed for simulation could be implemented.

At the end of the graduation it was possible to simulate the final product by utilizing modules for: speed control, position of the car and more importantly network communication between cars. By creating graphical interfaces to show the current status of the car and the current position of the cars in the network, it is easier to see how cars react on each others messages.

Although network communication and simulating a real car is possible there’s still some room for improving the software application. During network communication a lot of data is being sent between cars which can cause network problems (overflows e.g.) when a lot of cars are within each others range. Also is the algorithm that is used in the software application not as advanced as it will be in the Connect & Drive project.

These improvements can however be done when the final algorithm and network implementation is ready. By building the software application dynamically and by using modules it is easier for other software engineers to continue with this project implement those changes.

Glossary

Abbreviation	Description
ACC	Adaptive Cruise Control.
Agile	Software development method.
C#	Programming language.
C++	Programming language.
C-ACC	Cooperative Adaptive Cruise Control.
CLI	Command Line Interface.
CPU	Central Processing Unit.
Cygwin	Linux like environment for windows.
DLL	Dynamic Link Library.
FDD	Feature Driven Development, Agile development method.
GUI	Graphical User Interface.
GPS	Global Positioning System.
HMI	Human Machine Interface
IDE	Integrated Development Environment
Java	Programming language.
LOS	Line of sight.
Microsoft Winsock	Standard Microsoft implementation of sockets.
MSDN	Microsoft Developer Network.
NS-3	Network Simulator 3
OS	Operating System.
SME	Small and medium-size firm.
SNT Qualnet developer	Scalable Network Technologies Qualnet developer.
TraNS	Traffic Network Simulator.
VANET	Vehicular Ad-hoc Network environment.
WiFi	Wireless Fidelity, used as a wireless communication protocol.
Winpcap	Windows library that utilizes the possibility to capture network traffic.
Wireshark	Network traffic capture software, previously known as Ethereal.

1 Introduction

Nowadays it is almost normal to get stuck in a traffic jam. To solve this problem a couple of actions can be taken but most of them are time or cost consuming and not wanted by local or national governments. To solve the problem of these traffic jams the Connect & Drive project is instantiated.

The Connect & Drive project itself is based on utilizing network communication between cars to control the speed of cars. When the speed is controllable, usage of the road can be optimized and it makes it safer than it is now. The Connect & Drive project is a collaboration of four companies, a research centre and three technical universities. Fourtress BV is one of the four companies and is responsible for the software that takes care of the communication and makes sure that the speed of the car is controlled.

Although the project has not been started too long ago and the hardware is not yet available, Fourtress wants to start developing the software so they can test their software already. The main reason for this earlier testing is keeping ahead of possible problems that may occur when the hardware is available.

To solve this problem, Fourtress is looking forward to have some kind of software that already has the functionalities of the final hardware. With this software they can start testing for possible unexpected problems and by making the software dynamically it will be easier to adapt to the real hardware.

During the graduation period the software that is able to simulate the final hardware and test software will be made. A step by step plan is made to get to the final goal with a clear view of the different phases. In this report these phases and progress is described.

After this introduction there will be some information about Fourtress in the second chapter. In chapter three the actual assignment is described more precisely. The knowledge and research needed for the assignment are stated in chapter four and five. Chapter six contains the information about the implementation of the knowledge and research where chapter seven describes the results of the implementation. The conclusion and recommendations can be found in chapter eight.

2 Fourtress BV

This chapter contains information about Fourtress BV, how it started, what they do nowadays, business-models and more about their 'in-house-projects'.

2.1 General

Fourtress was founded nine years ago (2000) by Harry Schlatmann and Pim Grol. They saw that companies wanted to invest in technology but did not want to take all the risks. To solve this they decided to step in and share the risk. Fourtress provided the software engineers but also took care of the risk for the new technology. This was a great advantage for companies to explore new technologies and for Fourtress to expand their knowledge.

Fourtress's core-business is deploying their software engineers as consultancy agents at customer's locations, in the field of embedded software and technical automation. This is the biggest part of all the Fourtress employees. Other employees, managers, mentors, trainees, graduates and software engineers that are not working at a customer location are located at the office.

At the office these software engineers work on so called 'in-house-projects'. These projects are either commercial projects or started by Fourtress for expanding their knowledge and technologies.

2.2 In-house-projects

As mentioned in the previous paragraph Fourtress also has 'in-house-projects'. These projects can be divided into two different categories. The first category are commercial projects, the second category contains projects which fit in 'a place called tomorrow' which will be explained below.

2.2.1 Commercial projects

When Fourtress works on commercial projects those projects are also part of the 'in-house-projects'. These projects can have all kinds of purposes. One of them is Videology, a project which aims at improving the usability of security systems.

This project's main goal is to create a standard for communication between the cameras and the hardware which captures the data. To achieve this, an embedded device is placed at the camera side and also at the receiver side. At the camera side the data is processed to see if the data needs to be captured or not. When the data needs to be captured, when a burglar comes in for example, a connection with the receiver is made and data is sent. However, when a security employee wants to look through the camera, he can manually send a signal from the receiver to the camera to start sending data.

Another commercial project of Fourtress makes it possible for elderly people to easily make video calls to friends or family. The main goal of this project is to keep the elderly people active and show them how new technologies can help them in their life.

2.2.2 A place called tomorrow

A place called tomorrow is an overall project which is build up on all of the 'in-house-projects' started by Fourtress for expansion of knowledge and technologies. The main goal of 'a place called tomorrow' is to demonstrate people what new technologies can do to improve the comfort of living in personal care, domotica and automotive.

A good example of a project which is part of 'a place called tomorrow' is the 'lunchbox e-wallet'. It uses fingerprint recognition as a payment option for the lunch. Before you start lunch, you simply scan your fingerprint and when it is recognized it deducts the cost of the lunch automatically from your salary. With this system in use you do not need to carry any money with you, you just pay with your fingerprint.

Another project that improves the comfort of living is the 'wireless ambiance' project. The main goal of the project is to create a personal ambiance in a room. This ambiance is created by adapting lights, music and images to the person that is in the room. By adapting these aspects the person will feel more relaxed.

2.3 Company structure

Figure 1 shows an overview of the company structure of Fourtress. This overview shows the different departments within the company. The software engineers, trainees and graduates are all brought together as 'Technical Employees'. They are supervised by the 'Technical Manager'.

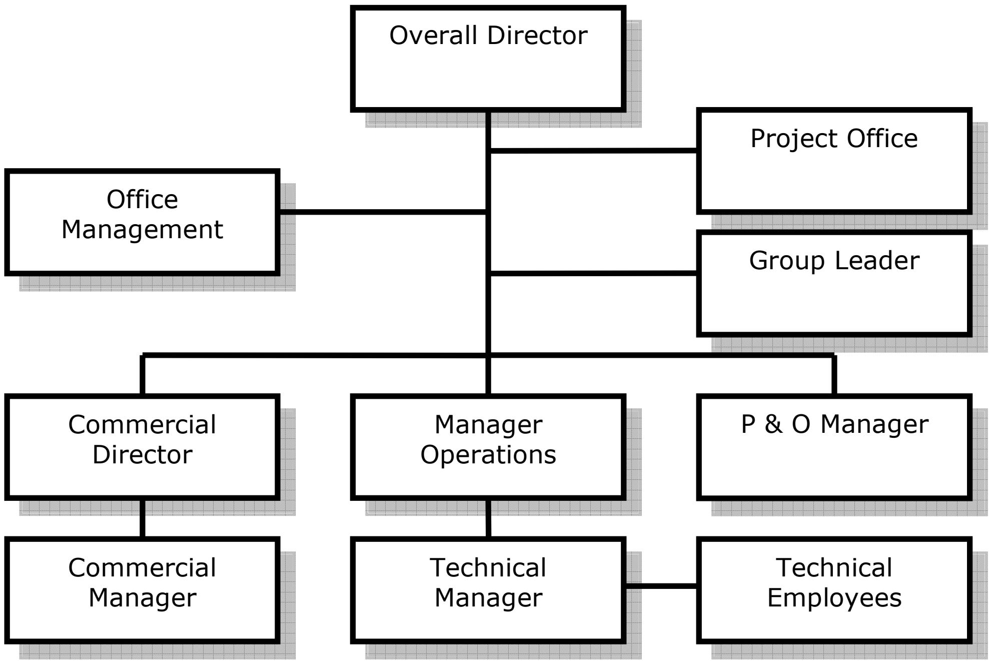


Figure 1: Fourtress BV. Company Structure

3 The assignment

This chapter describes the assignment, which is a part of the Connect & Drive project and which is described in paragraph 4.1. The assignment description itself can be found in the next three paragraphs. At first the initial situation is explained followed by the final goal of the assignment. In the third paragraph the project description can be found.

3.1 Initial situation

Connect & Drive is a project which includes all kinds of technologies needed to be taken car of, for making a successful product. During the Connect & Drive project all of the cooperating parties need to work together to complete the project. To do this, each company works on their part of the project.

The main part of Fourtress during the Connect & Drive project is making the software application that connects the different aspects of the overall project together. All of these aspects can be seen in figure 2, which also shows that Fourtress is in the middle of it all making the software essential for the Connect & Drive project.

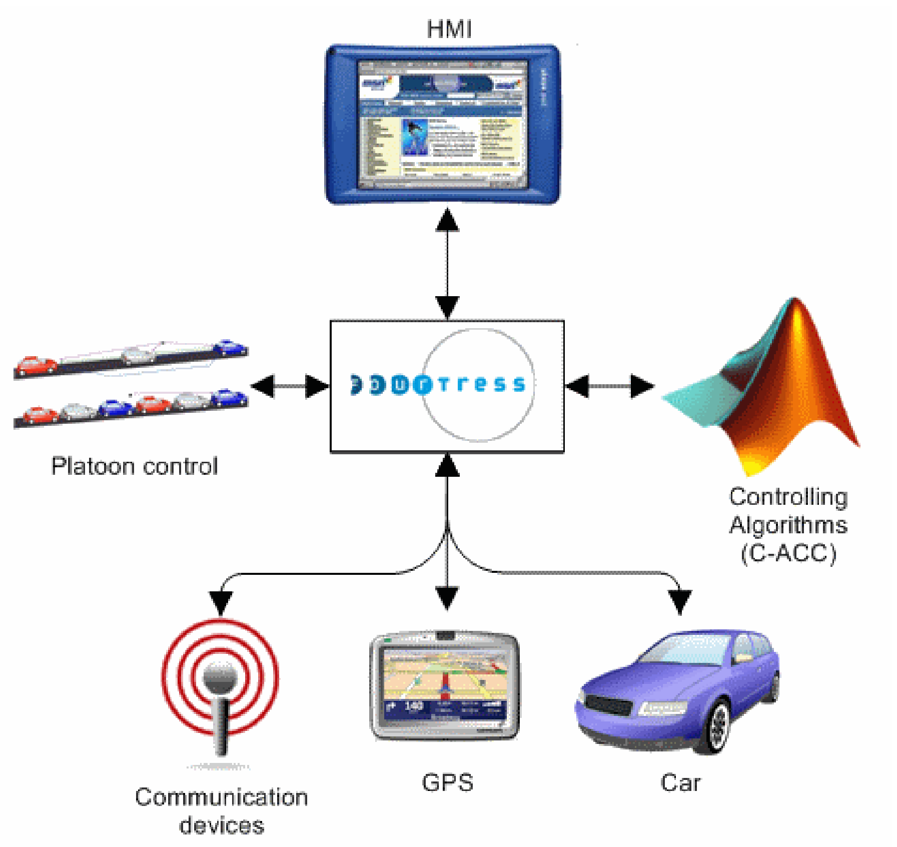


Figure 2: The overall technology part of Connect & Drive

As the figure shows the software needs to connect these hardware parts:

- HMI
- Platoon control
- Controlling algorithms
- Communication devices
- GPS
- Car

The first hardware part is the HMI, it can be a graphical interface as pictured in the figure but it can also be a red warning light inside the car. The platoon control is used to create platoons of cars already working together as a platoon on a road. The third important hardware part is the controlling algorithms. They calculate what the best to use speed is in case of nearing a traffic jam, bad road conditions or possible other hazards. The controlling algorithms are also based on communication that is received through the communication devices and the GPS. In the end, all these hardware parts are placed inside the car.

As the figure makes clear all the different hardware parts are connected to each other by software. The most important reason for this is that the hardware can only do what they are supposed to do. The GPS system can for example only determine its own position and not the position of the car ahead.

For the software to work completely as it should be the hardware components need to be ready before the software can be implemented. The reason for this is that cars can not use the C-ACC algorithm without communication for example.

3.2 Final goal

For Fourtress it is important to look towards the final goal of the project. They want to know what the system should do in different situations, how their software should respond to changes from the hardware and with this knowledge work ahead to prevent delays in the end of the project.

With this in mind, the final goal of the assignment can be described. The final goal is to create a software application that can act as a car, equipped with the final product of the Connect & Drive project. When the software application is ready it should be easy to modify it to changes in hardware and it should also be easy to test it in a real life environment.

To act as a fully operational system the software application needs to use all off the hardware parts that are shown in figure 2. But because Fourtress needs to work ahead and already start with some testing they need to simulate the parts that are not yet available. This simulation will be done by creating a stub which then can be used for controlled testing. The parts that are needed for the C-ACC system to work as it should be are:

- Network communication.
- Speed control.
- GPS.
- C-ACC algorithm.

Each part that needs to be simulated is similar to the real version of the product. This makes it possible to develop a software application and when the real parts arrive at Fourtress they can be implemented without having to change a lot or even start all over. When the software application is ready and a fully operational system is made, it needs to be tested.

To test the systems behaviour, a test situation as drawn in figure 3 should be possible. It shows a car that wants to join the traffic with cars passing by on the highway. The system should be able to determine the position of its own and use that in combination with its speed to calculate a safe speed to join the traffic. To calculate this speed communication with other cars is needed.

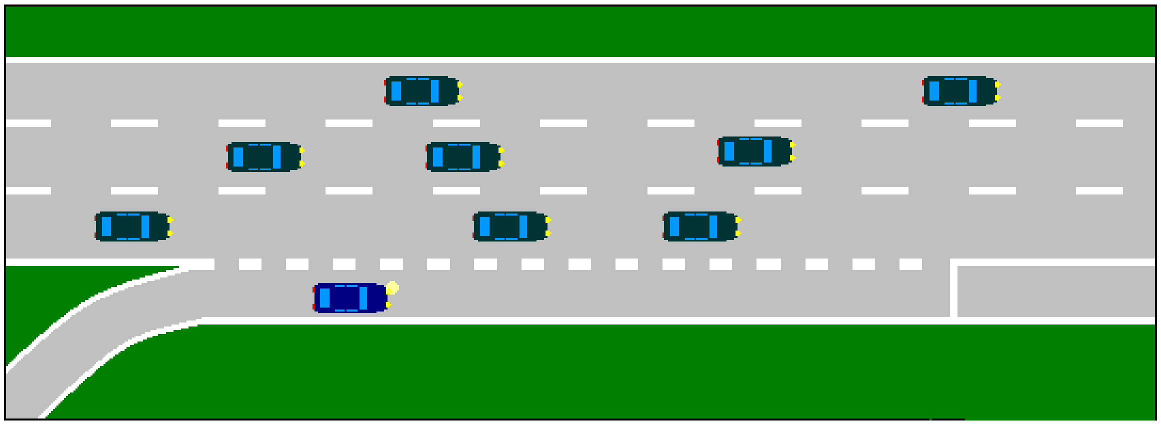


Figure 3: Car joining the traffic

During the development of the software application, Fourtress does not want to spend a lot of money for testing each time a new version of the software is available. To solve this problem and still be able to test the software, another software application must be made.

This software should make it possible to listen what cars communicate with each other over the network and display the information in a graphical interface. This enables the software engineers to test minor changes more easily. Because of the graphical interface it can also be used to demonstrate the effects of the C-ACC to visitors and customers of Fourtress, but it can also be used during presentations or conferences because it gives a nice demo effect.

3.3 Project description

Since the initial situation and final goal are clear it is possible to define the project description. The project description is the step to get to the final goal from the initial situation. What this step really means is explained in this paragraph. Because it is a big step to take, the step is separated into five objectives. These objectives can be found after the structure of the project is defined. This structure shows who is responsible for what during the graduation project.

3.3.1 Structure

During this project the following project structure is used to make clear who is responsible for what part of the objectives. This project structure is used on most of the ‘in-house-projects’ at Fourtress. The structure can be found in figure 4.

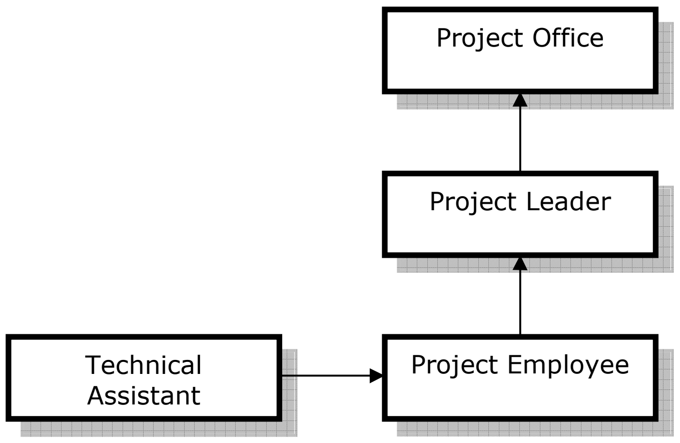


Figure 4: Project structure

As you can see in the figure above, the project employee justifies its actions to the project leader and gets technical assistances by the technical assistant. The project leader also assists the project employee by advising in situations where the employee is not experienced enough. Another task of the project leader is to report the status of the project to the project office. The project office keeps track of all projects that take place at Fourtress.

During this project the following people will take care of the different project roles. The project employee of this project is Ruud van Heugten who is a graduate at Fourtress. He gets technical assistance of Bruno Peeters. All of the work that is done by the project employee is reported to Niels Vos as the project leader. He will also help with making decisions and tries to stimulate the graduate to a better overall product. The project office is only has a small part during this project, its main task is to maintain the overall process and make sure the project stays on track.

3.3.2 Objectives

To make the step from the initial situation to the final goal a bit easier the step is divided into five objectives. Although these objectives are clear they are still too big to complete in one single step. To make these steps smaller, sub-objectives are introduced. These sub-objectives are smaller steps which are easier to accomplish but still provide enough improvement to finish the bigger objective.

Network communication
The first objective is to make network communication possible. Network communication is an essential part of the Connect & Drive project and also the main goal during the graduation period. The cars need to communicate with each other over the network to warn them for possible hazards, traffic jams and send them other messages regarding road safety.

- Message from client to server:
  The first sub objective is to send a message from a client application to a server application over a network connection. This is the basic principle of network communication. When messages can be sent from a client to a server, the server can be expanded to make it possible for multiple clients to send messages to the server application without causing interference on the server responsiveness.

- Respond to a specific message to specific client:
  When it is possible for the server application to receive messages from multiple clients at the same time, it needs to respond to those messages in a proper way. To

implement this functionality it can be easier to implement it first in a single client-server-environment. When server application is able to respond to the messages of this single client it can be expanded to support a multiple client-server-environment.

- Integrate clients and server:
  The overall final objective is to have one single software application that can do everything that is needed for the Connect & Drive project. To already work towards that goal the client and server applications need to be combined to one single application. After they are combined together the same functionality as in the previous sub objectives should be present.

- Make it all dynamic:
  After clients and server are integrated in to one application, the application should work without too much configuration needed before you can send and receive messages. To do this another sub objective is introduced to make sure everything works dynamically, without user interaction.

Speed control
To make sure that cars, equipped with the software application, drive the correct speed the speed control objective is necessary. The main goal of this objective is to know the current speed of the car but also control the speed of the car. This control is necessary when the car needs to brake or in situations where the car could speed up.

- Determine current speed:
  Before it is even possible to adjust the speed of the car, to for example other cars or hazardous situations it is necessary to determine the current speed of the car itself. With this speed it is possible for the C-ACC algorithm to calculate in how many seconds the car is going to be at a specific position.

- Adjust speed:
  Because speed is an important part of the Connect & Drive project, the ability of adjusting the speed of the car is also very important. In this sub objective this ability will be implemented.

Positioning system
During communication between cars it is important to know where the cars are. Not only to determine how much distance there is between cars but also to know on which road they are. This is very useful in the C-ACC algorithm because when cars join the traffic it needs to be communicated to other cars on the specific road. The positioning system will also be used to know in which direction cars are driving.

- Determine current position:
  For the system it is important to know on what road the car is. This is not only necessary for the C-ACC algorithm when a car is joining the traffic, but also to determine the best route when other cars communicate that a traffic jam is near. Other possible usage of the current position is when restaurants want to advertise to cars in a specific area.

- Determine driving direction:
  In a lot of situations it is not necessary for cars to communicate with cars driving in the other direction. If you know the driving direction of a car you can save time on communication and connection build up. The driving direction can also be used to warn the driver if he is driving the wrong way on a one way street.

C-ACC algorithm
After the network communication, speed control and positioning system are ready the C-ACC algorithm can be created. It utilizes all the previous objectives together and forms an algorithm that adapts it speed with information it receives through network communication and the positioning system.

- Create a ACC algorithm:
  The C-ACC algorithm is a difficult algorithm that needs to take care of a lot of different influences but the ACC algorithm is easier. Therefore this algorithm is implemented first to test out in an early stage how the rest of the system reacts. If everything works without problems it can be expanded in another sub objective.

- Define what information is needed for expanding the ACC algorithm:
  Before the ACC algorithm can be expanded it is necessary to make clear what the C-ACC algorithm needs to work properly. This is another step which is important for the further implementation of the C-ACC algorithm.

- Expand the ACC algorithm and make it a C-ACC algorithm:
  When it is clear what the ACC needs to make it a C-ACC algorithm it can be expanded. This is the last step for the main part of the final product. This means that it needs to be tested very well before it can be used in a real car. The testing however will be done in the end objective.

End objective
When the C-ACC algorithm is finished, the end objective is to implement everything in one system that can be installed in a real car. To do this, every objective needs to be tested on its functionality and if they do their work properly everything can be tested together. When these tests are done, the system can be tested in a real life situation. These tests can only be done when the hardware is available.

- Combine everything in a working test application:
  The first sub objective of the end objective consists of combining all the different parts from the previous objectives. When all the different parts are combined they need to be tested very well before they can be implemented in a real car. The best and easiest way to do this is, is by creating a test application. This test application will act as a real car but only on a computer.

- Create a monitoring program for all the cars:
  Although a test application is made for the cars itself, it is important to see where the cars are. This is done by creating a monitoring program that captures all the network communication done by the cars. After capturing the network communication the application analyses the data and transforms it in to a graphical interface which makes it easy to see where cars are and at what speed.

- Implement the software in a real car:
  When tests with both the test application and the monitoring program are successful it is time to implement the software in a real car. After implementation a series of tests is done to make sure the software will not cause unwanted situations on the road.

4 Knowledge

Knowledge is a necessary part of getting a problem solved. The problem can be solved with knowledge which is already at hand or acquired during the problem solving. During the assignment some knowledge was already at hand but also a fair amount of knowledge was gained. In this chapter the various parts of knowledge needed during the assignment are described. The first paragraph explains some general knowledge needed for the project to begin with. The other paragraphs tell more about the personal knowledge needed.

4.1 Connect & Drive

The graduation assignment was part of the Connect & Drive project. This chapter explains more about the Connect & Drive project.

Connect & Drive is a big project in which four SME, one research facility and three technical universities work together towards a common goal. This goal is to create a Cooperative Adaptive Cruise Control (C-ACC) system based on WiFi communication between cars and infrastructure. With this type of cruise control you are able to tell and listen to other cars about possible hazards ahead. Reducing the longitudinal wave motion in traffic jams is however the main reason for the communication. You can also use the communication to receive information about road closures or even advertisements of local firms.

4.1.1 Background

The current society is very dependent on mobility, although it causes severe problems on safety, congestion and the environment. In the Netherlands, road safety is better than in most other countries around the world, however it lacks in the flow through of traffic. The national traffic intensity is almost nowhere as high and congestion has become a national problem even with a good road-system.

The 'Kennis Instituut voor Mobiliteitsbeleid' reported an increase of 53% in travel time loss between 2000 and 2006 in the Netherlands and after 2006 it even got worse[1]. The main problem is: when more people want to travel, the worse it gets. So when the need is the highest the flow through is the worst.

To reduce traffic jams a lot of investments are done in increasing the capacity of the road-system. However, construction of more roads often is not what residents living nearby want. It is also expensive, takes a lot of time to finish and it is also bad for the environment. These are the main reasons to solve the problem by using the available roads. This can be done by spreading traffic better over the day and let the traffic make more use of the local roads instead of only using the highways. Also the public transportation can help reducing the amount of cars on the highway. Although reducing the number of cars will not increase the capacity of the overall transportation system.

By using innovative technologies the capacity of the current road-system can be increased. This is what the Connect & Drive proposal is all about. Focus on a new innovative technology to create new solutions. Connect & Drive wants to use C-ACC as a new Dutch technology to solve a big problem that is nowhere else as big as in the Netherlands. When the systems are ready they can be tested and implemented in a real good test-environment like the Dutch transportation system.

1: Mobiliteitsbalans 2008, Kennisinstituut voor mobiliteitsbeleid.

4.1.2 Traffic jams

As an effect of instability of the traffic stream, the infrastructure is used limitedly. Traffic jams are caused by human shortcomings, natural human driving behaviour is that people brake just a bit harder than the car in front of them. This causes a negative oscillation effect in traffic flow through. In a specific traffic flow this causes a longitudinal wave motion over the highway. This means that at one point you are standing still and a just minute later you can drive without problems.

To improve throughput, it is necessary to reduce the negative oscillation caused by human behaviour. This can be achieved by explicitly advising the driver the optimal driving speed and perhaps take control of it. A good progress in this process is the Adaptive Cruise Control (ACC).

ACC utilizes a cruise control system which bases its speed on distance to the car ahead and the speed of the other car. With these two factors not only the comfort of the driver is improved but also the safety. However, because ACC can only see the car in front and not further, it can cause to even exaggerate the wave motion when long lines of cars all use ACC and the first car abruptly brakes.

4.1.3 Cooperative Adaptive Cruise Control

Connect & Drive describes a research for a new generation in-car-system technology. This technology is called ‘Cooperative Adaptive Cruise Control’ because the cruise control works together with other cars and adapts itself to new situations. The main advantage of C-ACC over ACC is the communication.

With this communication it is possible to tell and listen to cars further ahead or behind you. These cars can help you determine what speed is the best to use for example. This speed can be calculated on possible traffic jams ahead but also weather or road information which you cannot see yet. The wave motion normally caused by natural behaviour can be limited because you already know what is going to happen. Another good reason to communicate between cars is when a car wants to join the traffic. When cars tell each other how fast they should drive and with which distance from each other it is safe and easy to join in.

In order to use a system like this, it is necessary to control all the different aspects of it in an easy way. In the Connect & Drive project each aspect is done by its own specialist. The different aspects are for example: controlling the speed of the car, communication between cars, check weather/road conditions and many more. To control these parts, a software application needs to be made. For the Connect & Drive project Fourtress will take care of that. Their software sends signals to the car to slow down when they receive a bad-weather-message for example.

4.1.4 Intended result and goals

The overall intended result is to show that, the newly developed technology in combination with several traffic-control-techniques, can shorten the travel-time on known traffic jam-routes with about 25%[1].

A set of goals is created to act as a base for the project. These goals are:

- Proof the current state-of-the-art ACC technology causes instabilities when a lot of ACC-cars are in line without car-to-car communication.
- By using an advanced decision-algorithm, stabilizing the oscillations of a (long) string of ACC-vehicles whom are teased by interferences on the follow-speed, follow-distance, changing numbers of vehicles and on non-communicating vehicles with the use of vehicle-to-vehicle communication.
- Realizing and justifying of trustworthy and robust WiFi communication technology to set up ad-hoc vehicle- and infrastructure-networks 'on-the-fly' which are strongly influenced by changing traffic density and mutual speed differences.
- Increasing/decreasing of the (follow) speed and follow distance in case of interferences with vehicle-vehicle communication and vehicle-infrastructure communication by using WiFi technology.
- Making clear what the impact of this in-car technology is for the flow through dynamics and utilisation of the (Dutch) road-system and solving/prevention of traffic jams.
- Identifying the success factors for the acceptation of C-ACC in both the system-driven and driver-driven system.

4.2 Methods

During the graduation period a couple of methods where used. One method was used to keep track of the process of the software development. To help with the software development a couple of design patterns from the object oriented software development method were used. This paragraph explains both methods starting with the Agile software development method.

4.2.1 Agile

During the project an Agile software development method was used. Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. This means that during iterations there will be a series of steps containing analysing, designing, implementing, testing and documenting. This makes it possible to improve the overall project in each iteration without introducing new problems.

The main problem people can think of when using Agile is that it causes a chaotic design and code. However, by analyzing at the start of iterations, the design and implementation of the features are done in the best possible way. Also by analyzing, refactoring of previous code is more likely to happen.

1: HTAS, Connect & Drive Projectplan.

The Agile software development method was not completely new at the start of the graduation period because the last project at school also used the Agile as software development method. Although it was not completely new, the process of the project at school did not went exactly the way it should be, because it was completely new. This caused some flaws which should not have to occur when using Agile.

During the assignment, the 'Feature Driven Development' (FDD) method was used. The most important part of this method is keep improving the project by focusing on new features, but also improving present features. This creates the need to keep analysing the previous project conditions. Because of this analyzing it is clear what can be improved or needs to be added to the project.

As said, each iteration also contains designing. As part of the FDD method used, the designing part is aimed at the feature. This causes to look for the best solution for the specific feature while keeping it simple. Implementing the design in the code will also be kept simple because it is not as big as a complete design. After implementation is done, the test-phase is there to make sure the feature implementation is done in a correct way. After all tests are positive, documentation of the next feature can be started and thus making it possible for the next iteration to implement the necessary improvements.

4.2.2 Object Oriented Design Patterns

Before starting the graduation some object oriented design patterns were already known. These had already been used at school and were implemented in some software applications during lessons. Although the knowledge was there, the usage of the patterns was not that clear. It also was not clear what the design patterns could mean for the design of software.

During the graduation period it quickly became clear that design patterns were needed to improve the dynamics of the software. This became clear when some refactoring needed to be done and a lot of important functions were tangled up in each other. To solve this, some learning was needed.

This learning made it clear that a good design with the use of design patterns can improve the design of the software application a lot. After this learning, the design of the software application needed a review. It was clear that it needed some drastic changes.

A big change in the design was done by implementing the Singleton pattern. The Singleton pattern is basically a pattern that makes sure you can only have one instance of an object. This makes sure you do not run into the problem of using the wrong object, although it does the same. The singleton pattern was used to provide a logging class in the software application. This logging class took care of all the logging that needed to be done. Because it was implemented as a Singleton class, making mistakes during the usage of the logging parts impossible.

Another pattern used during the graduation period was the adapter pattern. The adapter pattern is used to act as an adapter for a general interface. The adapter makes it possible to use the same interface over and over again without changing the design and implementation. The only part that needs to be changed is the adapter that makes sure the interface gets what it wants to have. The adapter pattern is used to maintain the same input and output functions but making it possible to use different input and output sources. Figure 5 shows the basics of the adapter pattern.

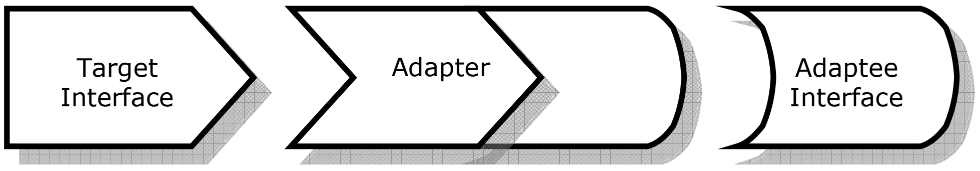


Figure 5: Adapter pattern

4.3 Programming

The programming language used for the software applications, made during the graduation period, was C++. The choice for C++ was quickly made. The main reason for it was: the knowledge that was already at hand. At school the main programming language was C++ and also a bit of C. Another reason to choose C++ is that it makes object oriented programming possible and most of the projects at Fourtress are also in C++.

The most important difference with programming during the graduation period was the difference in IDE. At school, projects programmed in C++, had to be made with Borland C++ Builder as the IDE. At Fourtress they do not use Borland C++ Builder but Microsoft Visual Studio 2005. Although it was a new IDE, it was not that hard to get started with it. The main reason being the extensive help files that come with it.

Although C++ is a strong language, it is not that dynamic to build nice and dynamic GUI's with. For this task C# was introduced to be using the same IDE as with the C++ programming. The only problem that came around was, when the C++ part needed to be integrated in C# by using dynamic link libraries (DLL). The reason for this is that the C++ DLL's are unmanaged and C# code is managed. More information on this subject can be found in paragraph 5.3.1.

Because of the dynamic qualities of C# it was not too hard to start programming. And even when problems occurred, the extensive developer network of Microsoft (MSDN) was there to help out. The MSDN also helped with problems related to the IDE and C++ programming in Microsoft Visual Studio 2005.

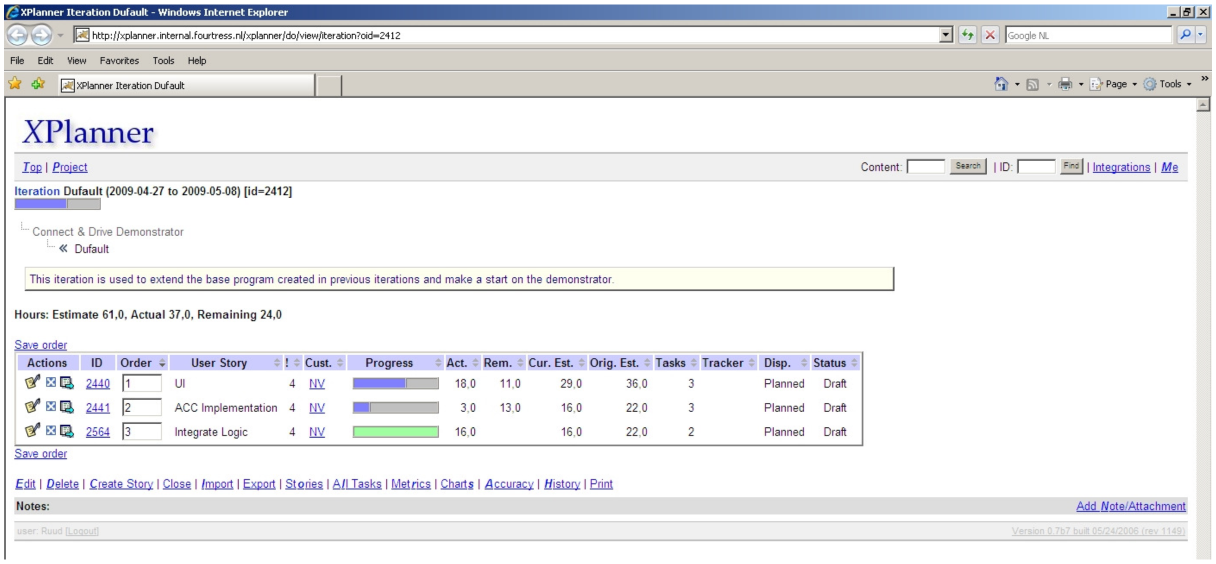
4.4 Project Planning

For the project planning at school most of the time the V-model was used. This caused that the overall planning was made up at the start of the project. There was a small possibility to change it further on but most of the time it was a strict schedule that needed to be followed.
During the last project at school Agile was used and made the planning of the project more dynamic. Although Agile stands for dynamic and iterative build up software developing, it does not mean you do not have to plan the overall project. However this was not clear on the school project and caused it to be an open end.

To make sure the assignment for the graduation period was not going to be an open end. The final goal and the steps to that final goal were made clear. With the use of Agile based on iterations it was made clear when what needed to be done. These iterations varied in length from one week up to three weeks. This dynamically build-up of the overall project made it possible to shift some tasks further back or start them earlier on. This shifting was done depending on the project progress.

Because of the dynamic build up it is even more necessary than usual to keep track of the progress in the project. This tracking was done by daily stand up meetings with all the software engineers. During these meetings a software engineer tells what he has done yesterday and what he is going to do the day of the meeting. He also tells what problems he encounters and thus gives other engineers the opportunity to help him fix these problems.

Although these daily meetings give a good indication for the process of the project it is necessary to document the process. To do this without causing a lot of overhead for the documentation Fourtress uses a software tool. This tool is XPlanner[1], an open source tool which provides a web-based interface with the ability of creating new iterations. After creation of an iteration it is possible to define objectives and for each objective create different tasks that need to be done by a specific engineer.


XPlanner Iteration Dufault - Windows Internet Explorer
http://xplanner.internal.fourtress.nl/xplanner/do/view/iteration?oid=2412
Google NL
File Edit View Favorites Tools Help
XPlanner Iteration Dufault
Page Tools
XPlanner
Top | Project
Content: Search | ID: Find | Integrations | Me

Iteration Dufault (2009-04-27 to 2009-05-08) [id=2412]

Connect & Drive Demonstrator
« Dufault

This iteration is used to extend the base program created in previous iterations and make a start on the demonstrator.

Hours: Estimate 61,0, Actual 37,0, Remaining 24,0

Save order

Actions	ID	Order	User Story	!	Cust.	Progress	Act.	Rem.	Cur. Est.	Orig. Est.	Tasks	Tracker	Disp.	Status
	2440	1	UI	4	NV		18,0	11,0	29,0	36,0	3		Planned	Draft
	2441	2	ACC Implementation	4	NV		3,0	13,0	16,0	22,0	3		Planned	Draft
	2564	3	Integrate Logic	4	NV		16,0		16,0	22,0	2		Planned	Draft

Save order

Edit | Delete | Create Story | Close | Import | Export | Stories | All Tasks | Metrics | Charts | Accuracy | History | Print

Notes: Add Note/Attachment

user: Ruud [Logout] Version 0.7b7 built 05/24/2006 (rev 1149)

Figure 6: XPlanner User Interface

As said, XPlanner provides a web-based user interface. This user interface is clear and the usage is very intuitive. This makes it easier for new software engineers to start working with it. Because it is easy to work with there is not much overhead but it provides a nice overview for iterations.

1: http://www.xplanner.org.

5 Research

At the start of the assignment not all of the information needed was clear. This made it necessary to do research. This research ranged from factual research (network technology e.g.) to knowledge research (dynamic link libraries e.g.). All the research that needed to be done is described in this chapter.

5.1 Network

An important part of the Connect & Drive project is based on the network part. Cars need to communicate with other cars and also with the infrastructure. This communication all takes place over the network. To get started with the project it was necessary to determine the possibilities of the network protocol that was going to be used. To determine what network gives the best options, research was needed.

This research was built up in a couple of phases. These phases consist of a facts phase, a test phase and an implementation phase. In the facts phase the main goal was to get the facts straight to build up a base for the test phase. During the test phase it was necessary to see if the facts were true by simulating actual situations in a simulator.

5.1.1 Specifications

The first research part of the project was to find out some actual facts about different networks. This meant getting to know the maximum range, speed, moving speed and for example at what frequency they operate. Below are the results of the research.

Network type:	**802.11 G**	**802.11 P**	**802.16**	**802.16 E**
Range	33-100 m	250 m	5 km fixed LOS	2 km
Duplex Mode	Half	Half	Time & Frequency	Time & Frequency
Simultaneously Connections	1 / channel	1 / channel	>100 / channel	>100 / channel
Maximum Data Transfer Speed	54 Mbps	6-27 Mbps	70 Mbps	40 Mbps
Expected Data Transfer Speed	10–25 Mbps	3-15 Mbps	20-30 Mbps divided by # users	20 Mbps divided by # users
Maximum Moving Speed	Not available	140 km / h	Not applicable	100 km / h

Table 1: Overview of network research

After the research was done it was evaluated to look for the best solution for the Connect & Drive project. For the network to work in the project it needed to be able to keep working on certain circumstances. The network should be able to keep a connection when cars travel with a maximum of 120 km / h and the connection should be steady for about 100 meters. The data transfer speed on the network is not as important because the messages that will be send will not be that large.

For the facts research this meant that there was only one good option left and that was the 802.11 P network. Its range is wide enough to maintain a steady connection and the network still operates when cars are driving 120 km / h.

5.1.2 Test

After the factual research on networks was done it was necessary to see if the results were true when the 802.11 P network was being used in a real situation. To test the real capabilities of the network, a network simulator was going to be used. Because there is a large number of available network simulator the most suitable simulator needed to be found.

To avoid too much delay because of research only three network simulators were tested, to see which simulator suited the best with the project and capabilities of Ruud van Heugten. These three network simulators were used in other projects done by other graduates. In the table below are the results of the research.

	Windows	Linux	Ease of use	Documentation	Interface	Possibilities	Cost		Total score
NS-3 Network simulator	1	3	1	1	1	1	3		11
TraNS	2	2	2	1	1	2	3		13
SNT Qualnet Developer	3	1	3	3	3	3	1		17

Table 2: Overview of pro's and con's of network simulators

- The ns-3 network simulator[1]: this network simulator is a discrete-event network simulator primarily for research and educational purposes. It is free software and runs on Linux and under Windows by using Cygwin.
- TraNS[2]: TraNS is a GUI tool that integrates a couple of network and traffic simulators. This makes it able to test VANET's and can be used on Linux and Windows because it utilizes the possibilities of Java.
- Scalable Network Technologies Qualnet Developer[3]: is ultra high-fidelity network simulating software. It contains an easy to use GUI which lets the user build up a network as it is in real life. This simulator can only be used under Windows.

During the test each network simulator was awarded a mark on each of the tested parts. This mark ranged from one up to three, where three is the best and one the worst. At the end all the marks were added and a final score came out.

Table 2 makes clear that the SNT Qualnet Developer came out as the best network simulator. The main reason is the good documentation combined with the clear interface and because it is easy to use. The downsides of this simulator are that it can not be started on a Linux OS and that if you want to use the full version of the program the software is not free. However for the graduation period an evaluation copy could be downloaded.

To get familiar with the program some examples were tested and tried to be reproduced. After a short while the program could be used in a good way to do some testing. The main goal of the testing was how the network could handle the signals being send by the cars. To test this, three groups of cars were simulated as figure 7 shows.

1: http://www.nsnam.org.
2: http://trans.epfl.ch.
3: http://www.scalable-networks.com/products/developer.php.

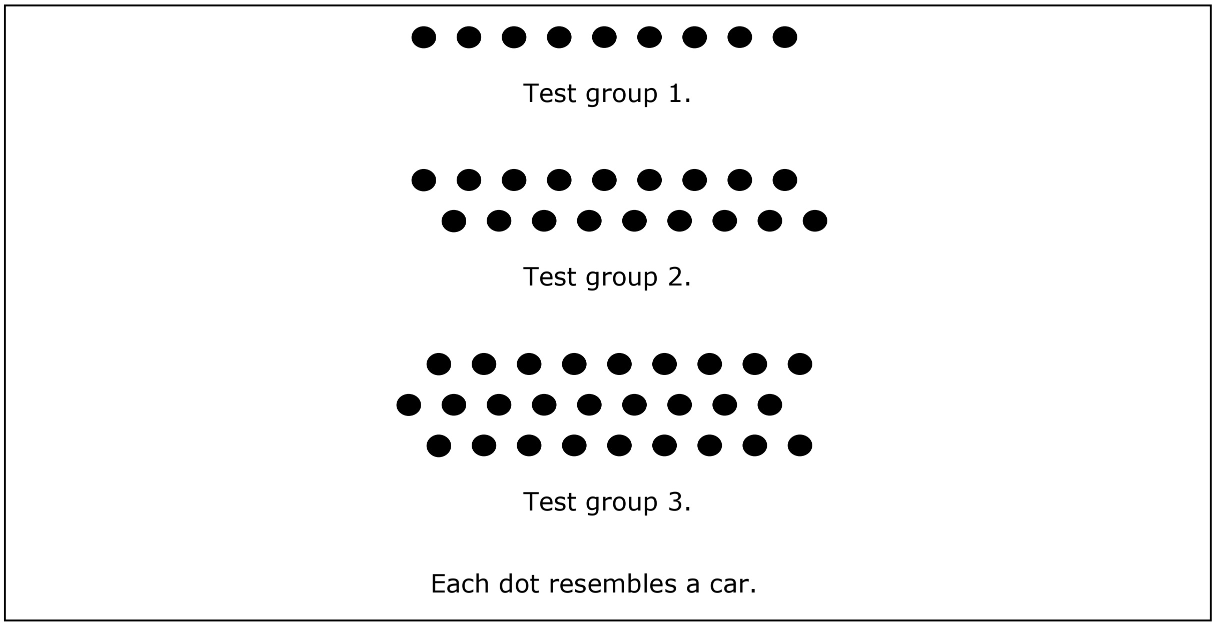


Figure 7: Test groups used in the Scalable Network Technologies Qualnet Developer

Each test group was tested on three tests. These three tests describe two, three or four cars communicating to each other. In each test the communication took place between cars that had the biggest distance between them. Each test took fifty seconds in which data was transmitted during thirty seconds.

Cars communicating:	**2**	**3**	**4**
Signals transmitted:	543	978	1729
Signals received and forwarded:	2194	4002	6828
Signals locked on:	3087	5737	9594
Signals received with errors:	893	1735	2766

Table 3: Test results of test group 1

Cars communicating:	**2**	**3**	**4**
Signals transmitted:	678	852	1246
Signals received and forwarded:	5895	8157	10916
Signals locked on:	7918	10361	14538
Signals received with errors:	2023	2204	3622

Table 4: Test results of test group 2

Cars communicating:	**2**	**3**	**4**
Signals transmitted:	861	1000	1664
Signals received and forwarded:	11681	12412	19682
Signals locked on:	15011	16663	26772
Signals received with errors:	3330	4251	7090

Table 5: Test results of test group 3

As you can see in the tables 3, 4 and 5 the number of signals drastically increases when more cars are near and more cars are communicating with each other. Because there are so many signals being send it is more likely that collisions of signals will happen. This will cause more signals not being received and have to be retransmitted.

5.2 Network communication

After basic network research was done it was necessary to get to know the details on network communication. These details consist of packets that are being sent between network devices. These packets contain information about the sender and the receiver of the specific packet. It also contains information about what flags are set in the packet and what data they contain.

To capture these packets and thus their details it was necessary to find a way this could be done. At first a program, which could do all of this, was searched and found in Wireshark[1]. Wireshark offers the ability to capture and filter packets that come across a specified network adapter.

However in the software applications it is necessary to capture and filter packets inside of the application itself. The best way to do that would be, using the same library that Wireshark uses. The used library of Wireshark is winpcap. Winpcap is open source, because it is open source it is available and thus can be used in our project. To use the winpcap-library some additional research was needed.

The library itself is documented very well and much information can be found online on its website. If there were problems which could not be solved with the provided documentation there is was a big user-base which was open for new users. This user-base is as big as it is because of the open source character of the library and none other packet-library is available for windows, thus being used in many programs.

The research on the usage of the library is split into a couple of sections which are explained in the next set of paragraphs. These paragraphs together form the total usage of the winpcap-library in the graduation-project.

5.2.1 Listening

To capture packets and see what information they contain it was necessary to listen on a network adapter. To listen on a network adapter it was necessary to open the specific network adapter. The winpcap-library offers a set of functions to complete this step. First of all a list of all available network adapters was used to select what network adapter was going to be used.

After the network adapter was selected to be used. It was necessary to open it. After opening the network adapter it is set in a specific mode. In normal mode the network adapter only captures the packets that are destined to it and thus not capturing packets that are exchanged between other hosts. However all packets on the network adapter needed to be captured. To do the network adapter needed to be opened in promiscuous mode. This allowed the program to capture all of the packets on the network adapter whether they are destined to it or not.

Once the network adapter was opened to capture packets listening could be started. Winpcap offers a couple of functions to capture packets. These functions can be found in the documentation of the library. Because it was not clear how many packets had to be captured the function which is capable of continuous capturing was chosen.

1: http://www.wireshark.org.

5.2.2 Filtering

After being capable of listening on the network adapter it pretty soon became clear that in a network environment, more packet-transfers are going on than you can imagine. Because we were looking for specific packets in our project. Filtering of all the packets was necessary. With the ability of filtering it was also possible to test or debug in a more detailed way.

The winpcap-library makes it possible to add a filter on the previously opened network adapter. To filter out all the unwanted packets it is necessary to compile the specified filter first. The filter uses a syntax which can be found in the winpcap documentation.

When the syntax of the filter is correct the filter can be set up so it is actually used by the capture function that is chosen. When a filter is set all the packets on the network adapter are filtered and only the ones that passed through the filter are captured.

In an environment where there is only specific data transfer between cars, filtering is still needed. Because when large groups of cars communicate the big number of packets being send could mess up your application. It is also helpful to filter packets that are not needed for a specific car.

5.2.3 Sending

Besides capturing and filtering packets another feature in the winpcap-library was used. This feature was the ability to send packets. This made it possible to respond in a specific manner on a captured packet. It is also used to look for other cars around and thus making it possible to set up new connections.

Another usage of sending out specific packets was to see how the application worked and reacted on simulated packets. This made it possible of simulating possible situations which can occur in real live but are too difficult or expansive, to test. Given the fact that these tests can be done, it is also possible to test the possibilities of the network hardware that is being used.

The winpcap-library gives two options of sending out packets. The first option is to send out one packet at a time and the second option makes it possible to build up a queue of packets and send the queue out at once. For the project it was not necessary to send out more than one specified packet at a time. Cars react on a received packet and that reaction needs to be sent immediately.

5.3 Programming research

While there was a good base knowledge of programming there was some additional research necessary on some topics. This research mostly consisted of testing and implementing examples to see how they actually worked.

5.3.1 Dynamic link libraries

As the program became bigger and bigger there was a need to have some parts gathered. This was possible because of the design had specific pieces, that took care of a specific task, were grouped together. These set of pieces could be changed in the future and it was necessary that they could be updated without having to recompile the complete software application.

The possible solution for this problem was the use of libraries. There are two general kinds of libraries: static and dynamic libraries. The main difference between these two is the time of linking them into your software application. Static libraries are linked at compile time and dynamic libraries are linked at run time.

Another difference between static and dynamic libraries is the use of it. A static is always embedded in the program and thus can not be removed or get corrupted. A dynamic library is apart from your program and may get corrupted or accidentally be removed by a user.

Because static libraries are embedded in the program, the program's size will be bigger and the program needs to be updated completely when the library gets code improvement. However a dynamic library can be update without the need of recompiling the program. It just loads the library and uses it without even knowing what changed. That was the main reason for choosing a dynamic library for our software application.

5.3.2 Application Programming Interface

During the software development some functions appeared in multiple classes. These functions were defined the same way but implemented in another way. This caused some overhead in code that was not necessary. To reduce the overhead and make the use of the functions easier, interfaces were being used.

An Application Programming Interface (API) defines a set of functions that can be used during implementation. Classes that implement these functions only need to make sure that the input and output of the functions are the same as the definition of the interface. This makes it possible to change between different implementations without changing the already compiled code.

Using an API can also be helpful for other programmers that are going to work with it. It reduces the time needed to look more deeply into the other code. During the implementation of an API time is reduced but it takes more time to create the API. So before choosing to use an API it is important to calculate what cost-effective wise the best solution is.

5.3.3 Threads

Threads are very helpful in situations where multiple processes are needed. Each thread has its own process and the process is executed without interference of the other processes. This makes it possible to execute a bigger process quicker than when not using threads. A good example is when there are three rooms that need furniture. In the first case there is only one employee that works as a decorator, a cleaner and as a designer. When taken into account that only one task can be done per day you will see the following progress for the three rooms:

	Room 1	**Room 2**	**Room 3**
Day 1	Decorated		
Day 2		Decorated	
Day 3			Decorated
Day 4	Cleaned		
Day 5		Cleaned	
Day 6			Cleaned
Day 7	Designed		
Day 8		Designed	
Day 9			Designed

Table 6: Example of one employee taking care of three tasks

Table 6 makes clear when room 1 is decorated it needs to wait two days before its being cleaned. By using threads the same three rooms would be done by three employees: a decorator, a cleaner and a designer. This would result in the following scheme:

	Room 1	**Room 2**	**Room 3**
Day 1	Decorated		
Day 2	Cleaned	Decorated	
Day 3	Designed	Cleaned	Decorated
Day 4		Designed	Cleaned
Day 5			Designed

Table 7: Example of three employees taking care of three tasks

This shows that dividing multiple tasks over multiple processes makes the it possible to finish tasks earlier and because more work can be done in the same time you can have multiple functionalities being used at the same time.

For the software application in the Connect & Drive project, threads were needed for the first time when it needed to listen and send messages at the same time. Because it could happen that the program was busy with sending a message and thus could listen for other messages. That could accidentally mean a loss of a message.

To use threads in C++ project there are three possible solutions:

- WinThreads: Microsoft provided threads that are included in Visual Studio.
- PThreads: a POSIX standard for implementation of threads, mostly used on POSIX like operating systems.
- Boost::Threads: Boost is a collection of libraries which also includes threads.

The choice made was based on the documentation that was available for each of the libraries. Boost offered the best documentation and without too much of a hassle threads were implemented in the software. When the threads needed some more specific tasks the documentation helped out with the problems that came across.

6 Implementation

During the implementation of all sub objectives, the overall software application and the design of the different applications changed. The most important reason for this is the use of the Agile development method. The changes made in the design show how the software applications progressed during the graduation period. This chapter describes the process of each of the five main software applications.

6.1 Server and client

The first implementation done for the project was creating a server and a client application. These two applications made it possible to send message over the network by using sockets. The use of sockets was not new and therefore a good start to begin with. These two software applications worked almost the same and figure 8 shows the sequence diagram of them.

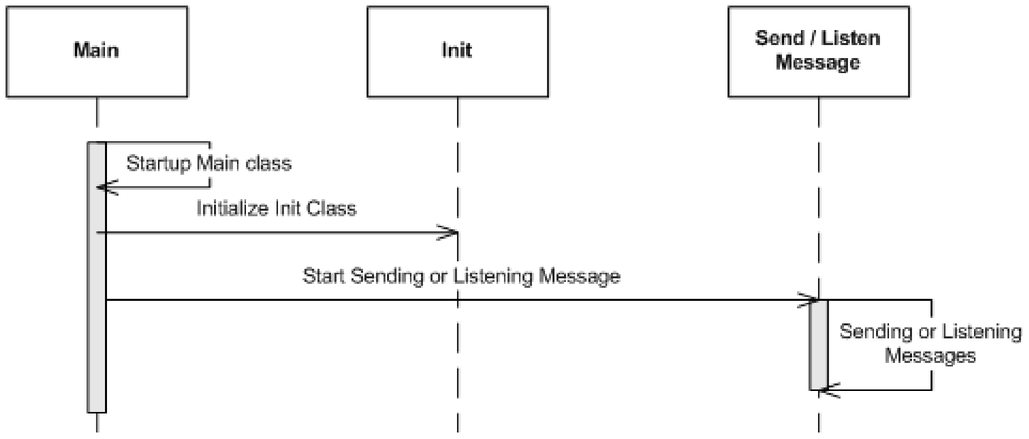


Figure 8: Sequence diagram server and client applications

Although these designs are pretty clear, they are not very useable for expanding the functionality. Because both software applications almost work the same and the designs are comparable the next application needed to integrate both the designs and applications.

6.2 ServerClient

The second implementation stage combined the server and client applications from the first implementation stage. The combining went well and the sequence diagram of the software application is shown in figure 9.

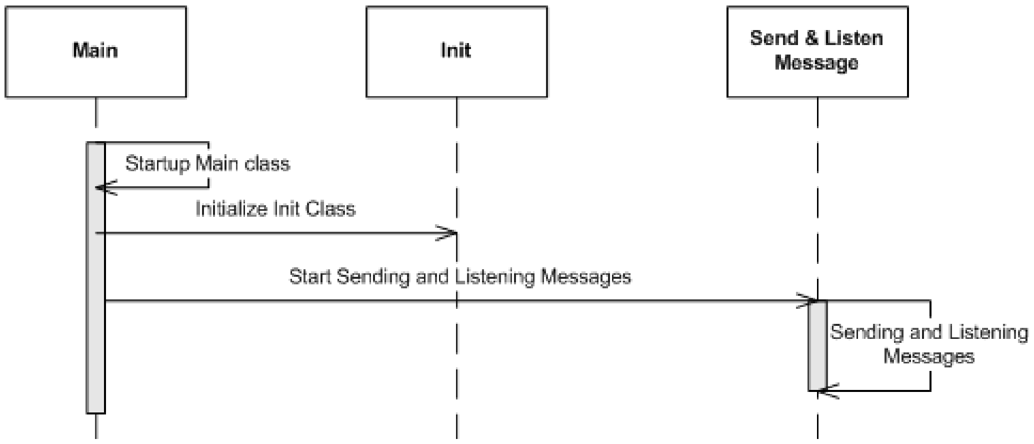


Figure 9: Sequence diagram for the ServerClient application

The ServerClient application worked the same as both the server and client of the previous implementation did. But problems occurred when multiple instances of the ServerClient were executed. Some instances received messages but some did not. To easily see what went wrong a logging class was introduced. The logging class wrote to a file to see at what time which code was executed. Although the logging worked well, it still was not clear what caused the problems.

With some technical assistance it became clear that the listenMessage function could not operate at the same time as the sendMessage function. To solve this problem, the functions needed their own thread to listen or send messages.

These threads did not change the design but now the program was able to send and receive messages at the same time without loss of messages. At that point of time the design of the software application was like the design in figure 10.

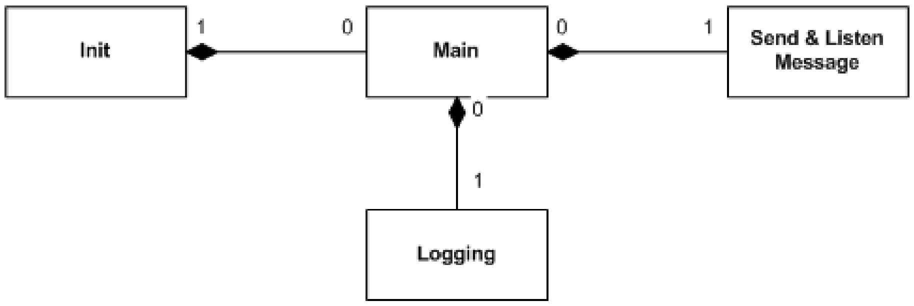


Figure 10: ServerClient design with integrated logging class

The basics of network communication was working for now but it only used sockets to send and receive messages but no detailed information on the packets was available. However, that detailed information was needed to exactly know what was going on during the communication. To work this out the winpcap-library was used and introduced in the design.

After some investigation to get a working library, the overall code was not really clear anymore and a lot of functions needed information from other functions to work. That was not a good situation and some redesign was needed. To make the design more clear, a lot changed and now almost every class took care of its own functions and values.

The only class that could not be left out was the logging class. It still was invoked by the classes itself. This caused the software to have multiple instances of the logging class. This became a problem when multiple classes tried to write to the logging file when another class was already writing to it. To solve this, the logging class became a Singleton class and could only be instantiated once. Figure 11 shows the final design for the ServerClient software application.

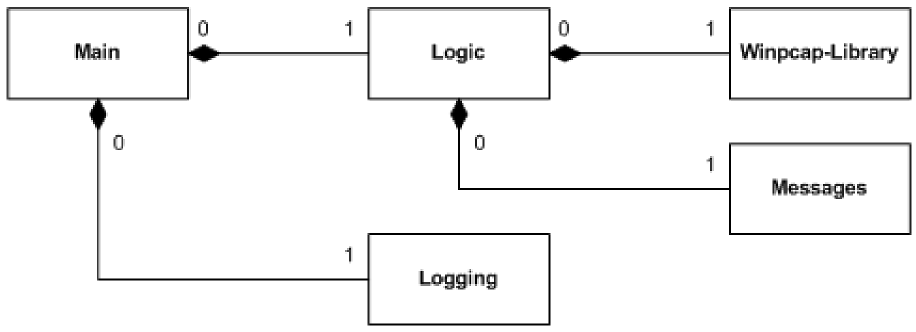


Figure 11: Redesigned ServerClient application

6.3 Car

The third implementation phase was focused on implementing the ServerClient application into a simulated car with the added ability of speed control and a positioning system.

The first step in this process was to change the names of the different classes that made the application as it was. The class names did not tell much about the actual functions and possibilities of the class. Although it was not a big programming step it was a big step for the readability of the design. An important part of the overall project was for other software engineers to understand and use the code afterwards and with the change of the class names this part was done in a better way. The design with the new class naming looked like figure 12.

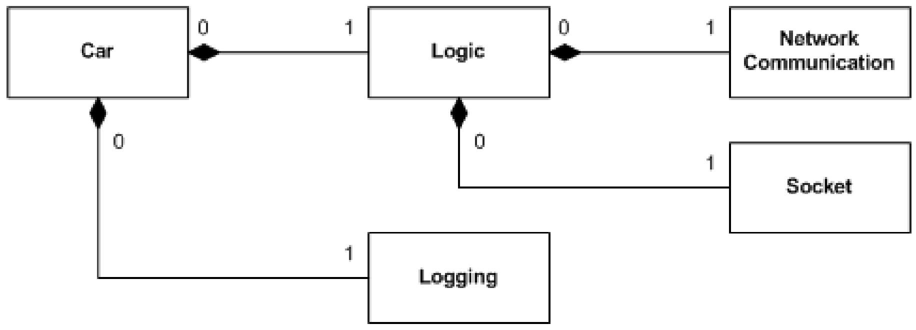


Figure 12: First Car design with renamed classes

With proper names it was time to start implementing the speed control and positioning system. Because at this time the final hardware for both the speed control and positioning system were not available, they needed to be simulated. To make the car even more like the final system, the socket class was removed. This was done because in the final system the only network communication is done by sending specific packets and not through socket connection. After the socket class was removed, the logic class was only used as a wrapper for the network connection thus causing code overhead. By removing the logic class the overhead was gone and after these changes the design of the car started looking like it should be in the final product. Figure 13 shows the design of the car so far.

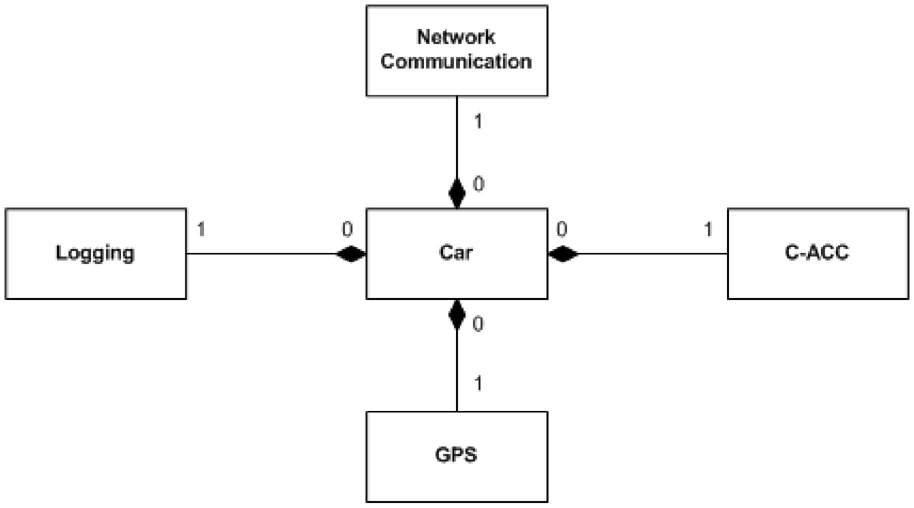


Figure 13: Car design with all needed classes implemented

With the speed control and positioning system, the car was almost ready to be tested in a test environment. But it still needed some more functions to determine the driving direction, the distance to the car ahead and the brake speed for example. These functions are the main parts of the C-ACC algorithm that also needed to be implemented. Another improved for the software application was a GUI. This GUI was especially useful to see the values of the various aspects being used.

To use the software application in a GUI, the easiest way was to embed it in a DLL. However, for the upcoming road application most of the car features were going to be used. To make the car more dynamic and gain the ability to use the different parts separately, all classes were transformed into DLL's. When the application was build up on DLL's, programming the 'car-DLL' could be started. A new software application was made which included the 'car-DLL' and only used the functions it needed in the GUI. This project than functioned as a visual wrapper around the car. After some more logical naming of classes the final design for the graphical car application looked like figure 14.

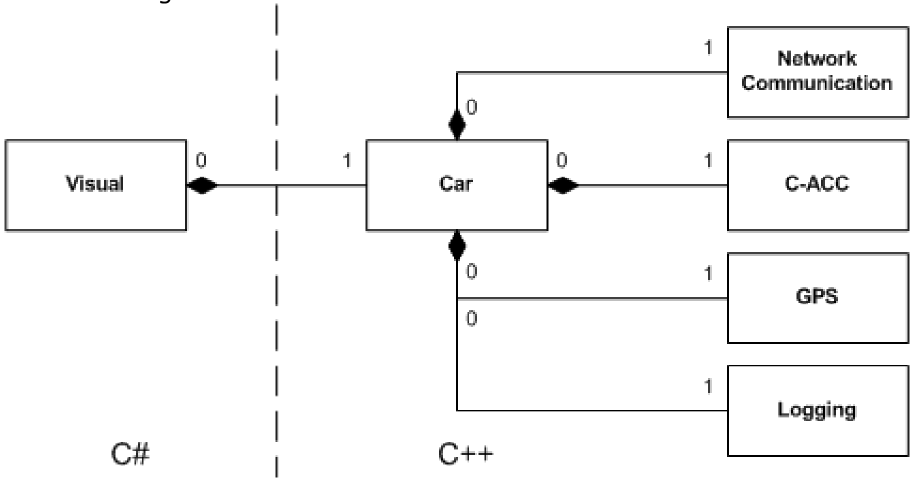


Figure 14: Final Design of the graphical car application

6.4 Road

After the car was made, it was clear that the system could not be implemented in a real car because of the lack in hardware available, to control the car. But to be able to show a couple of cars in a simulated environment it was necessary to create a software application that was able to do so. This software application was developed as a road that could listen to the network communication between cars and then use that communication to show the cars on a virtual road.

Because this road also needed to have the ability to capture network communication it would be easier to use the same class as used in the car. As said in the previous chapter, this was done by creating separate DLL’s containing the specific classes. For the road this meant that it only needed the network communication DLL and in a few simple steps the road was able to listen to the network communication. Because the road only needed the option to listen, a lot of functionality was stripped out of the DLL to decrease memory needed for the software application.

When it was possible to capture all of the network communication between cars, it was necessary to show this communication in a proper way. This was done the same way as with the car. Make an extra visual class which implements the methods of the road. The final design of the road can be seen in figure 15.

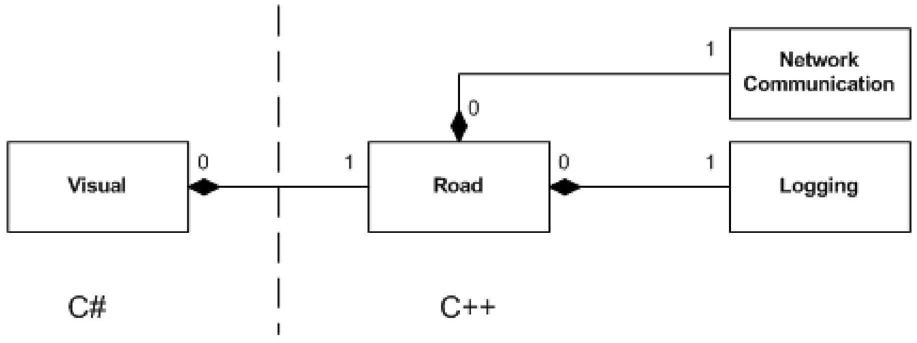


Figure 15: Final design of the graphical road application

6.5 Demonstrator

When the road was ready, the test environment was also ready and there could be multiple computers in a network, each equipped with the car application. The road was able to show the network communication between those cars. Fourtress also would like to have another application which made it possible to show multiple cars and a road on a single computer. To do this some additional coding was needed.

This additional coding consisted of combining both software applications and giving some more options to simulate a network. On this simulated network the cars can communicate with each other and the road can show the communication. To make it easier to use, one single application was made were the user is capable of creating a new car inside the simulated network environment. The design of the demonstrator is made out of combining the two designs into one design with a shared visual class. This design can be seen in figure 16.

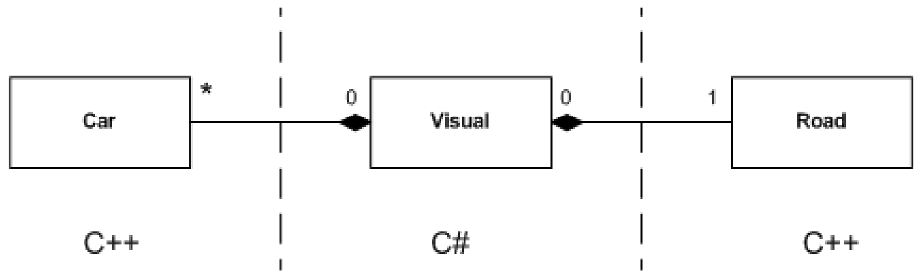


Figure 16: Final design of the graphical demonstrator application

As the figure above shows, a demonstrator can have multiple instances of a car and only one road instance. Network communication is utilized in both the car and the road environment but cars only communicate with other cars and the road only listens to the communication of cars.

7 Results

During the graduation period the overall progress of the software applications can be divided into five different sections. These five sections are the paragraphs for this chapter. In each paragraph will be discussed, what the main goal for it was and what result was achieved.

7.1 Server and client

The first part of the graduation period mainly focused on getting the basics of the network communication to work. This was done by creating a separate server and client application. The client application was able to send messages to the server which was listening for messages on the network adapter. After making it possible for a single client to send a message to the server, the client was expanded and that made it possible for multiple clients to send a message to the server. Figure 17 shows the message flow between the two applications. At this point only a CLI was being used.

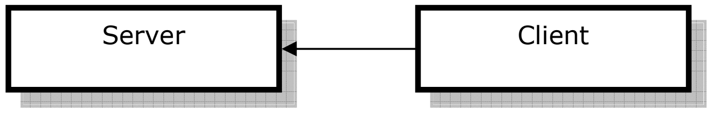


Figure 17: Server and client applications

Looking back at the objectives set at the start of the graduation period, there can be noticed that one sub objective is already finished which is: making it possible to send a message from client to server.

7.2 ServerClient

The second phase mostly concerned the improvement of the usability of the program. This included the use of only one software application instead of two. Another goal was to have the ability to respond to messages that were received. To only have one single software application, the previous server and client application were combined to provide a better usability. Besides better usability it also provided an easier way to implement changes and improvement to the program.

The other change implemented during this phase consists of responding in a specific manner to messages being received. That made it possible for a couple of application to start communicating and sending each other necessary information. At this stage there still was a CLI used as user interface. Figure 18 shows the status of the program so far.

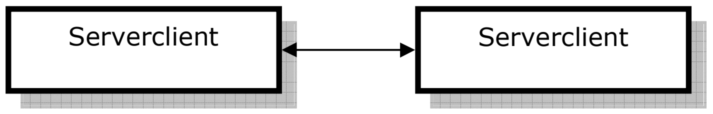


Figure 18: ServerClient application

During this phase two objectives were completely finished and one for a fair bit. The finished objectives are the objective of combining the server and client application being able to respond in a specific manner to a received message. The objective of making everything dynamic was partly finished because two systems could talk with each other but not completely without user interaction.

7.3 Car

After most of the network communication was done, a next step could be made. This step included the following changes: move from a ServerClient application to a car, change from a CLI to a GUI and implementing more basic functions towards the final goal. Because the step was that big, changes were implemented in a couple of phases to make it easier. At first the most important change was the design of the program. It changed from an unclear design to a well thought design. This made it possible to give a clearer view of the program and improved the possibility of future implementation.

The next step focused on making the car more real. This was done by adding the speed control and positioning system. These two systems were also very important for the C-ACC algorithm that needed to implemented, in this phase. But during this step it became clear that not all the objectives could be implemented the way it should be in the final software application of the Connect & Drive project.

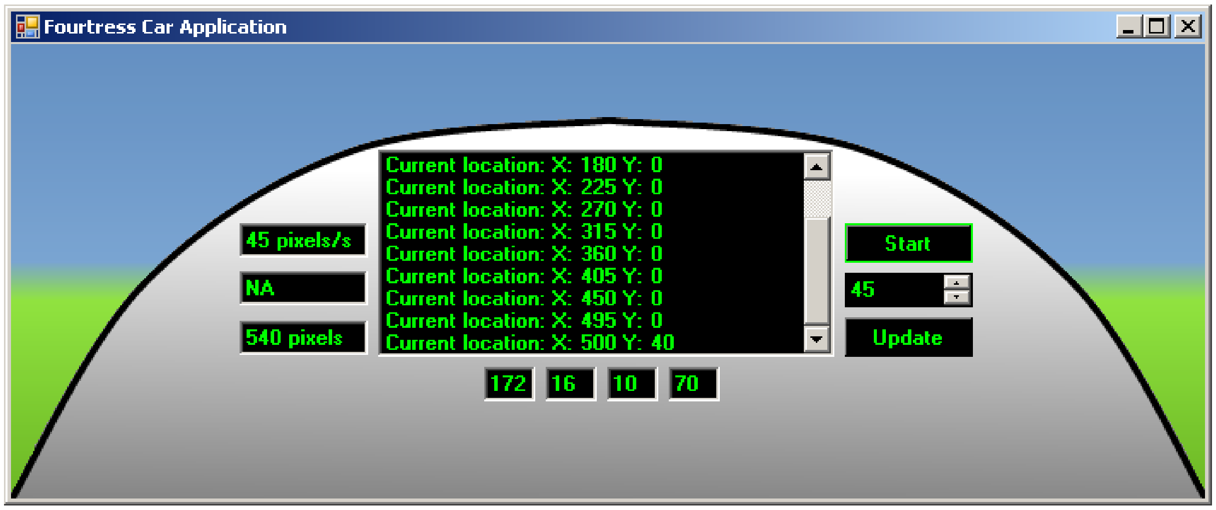
The necessary hardware to control the speed of the car and determine the actual position of the car was not yet available and thus could not be implemented. Although these functions of the car could not be implemented as they were planned to, they were simulated in the software making it still possible to create a good test environment for the cars.

After the cars were able to control their speed and their position, the communication between each other needed to be more dynamic. By making it more dynamic the car application can be used in a good test environment without too much user interaction. This was done by changing all parts of the network communication that were statically defined.

The last important step for the cars was, to implement the C-ACC algorithm. This algorithm makes it possible for cars to adapt to each other. At first an easier ACC algorithm was implemented to test if the communication between cars was alright. After tests showed that the algorithm could be expanded, by using the communication between cars to negotiate about their speed.

Although the cars were now able to dynamically build up a connection, communicate with each other and negotiate about their speed, the CLI still made the application a bit user unfriendly. To change this, a GUI was introduced which made all the used functions of the CLI visible in a nice and clear GUI. After this step had been done, the car was ready and could be used for testing purposes.

Figure 19 shows the final GUI used for the car.



```
Current location: X: 180 Y: 0
Current location: X: 225 Y: 0
Current location: X: 270 Y: 0
Current location: X: 315 Y: 0
Current location: X: 360 Y: 0
Current location: X: 405 Y: 0
Current location: X: 450 Y: 0
Current location: X: 495 Y: 0
Current location: X: 500 Y: 40
```

Figure 19: Car GUI

This phase finalized a lot of objectives. Most of these objectives are finished but only for the simulated environment. That is because the hardware that is going to be used in the car was not available at the time. That taken into account the following objectives were finished this phase:

Network communication:
- Make it all dynamic.

Speed control:
- Determine current speed.
- Adjust speed.

Positioning system:
- Determine current position.
- Determine driving direction.

C-ACC algorithm:
- Create an ACC algorithm.
- Define what information is needed for expanding the ACC algorithm.
- Expand the ACC algorithm and make it a C-ACC algorithm.

7.4 Road

When the implementation of the car was completed, work on the road application could be started. Refactoring the car to a road did not take too much time. Mostly because the main functionality of the road is: to listen to the car communication. This communication could already be captured by the cars. So getting the logic of the car to work for the road, required some refactoring of the cars.

It was necessary to create a nice looking GUI that was able of showing driving cars and displaying information about these cars. The most time was spent on testing how cars were shown and how the information could be shown better. In the end this resulted in the GUI as shown in figure 20.

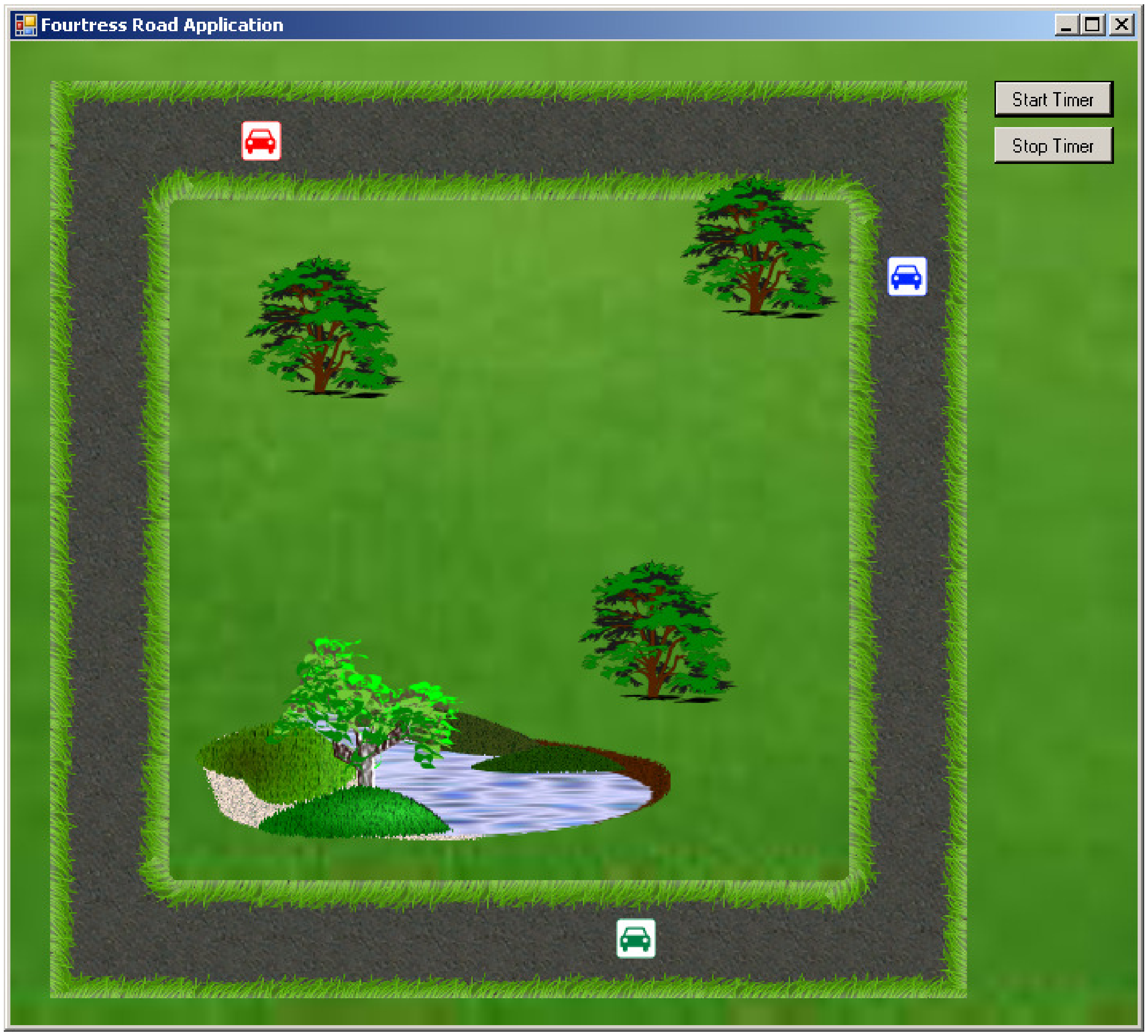


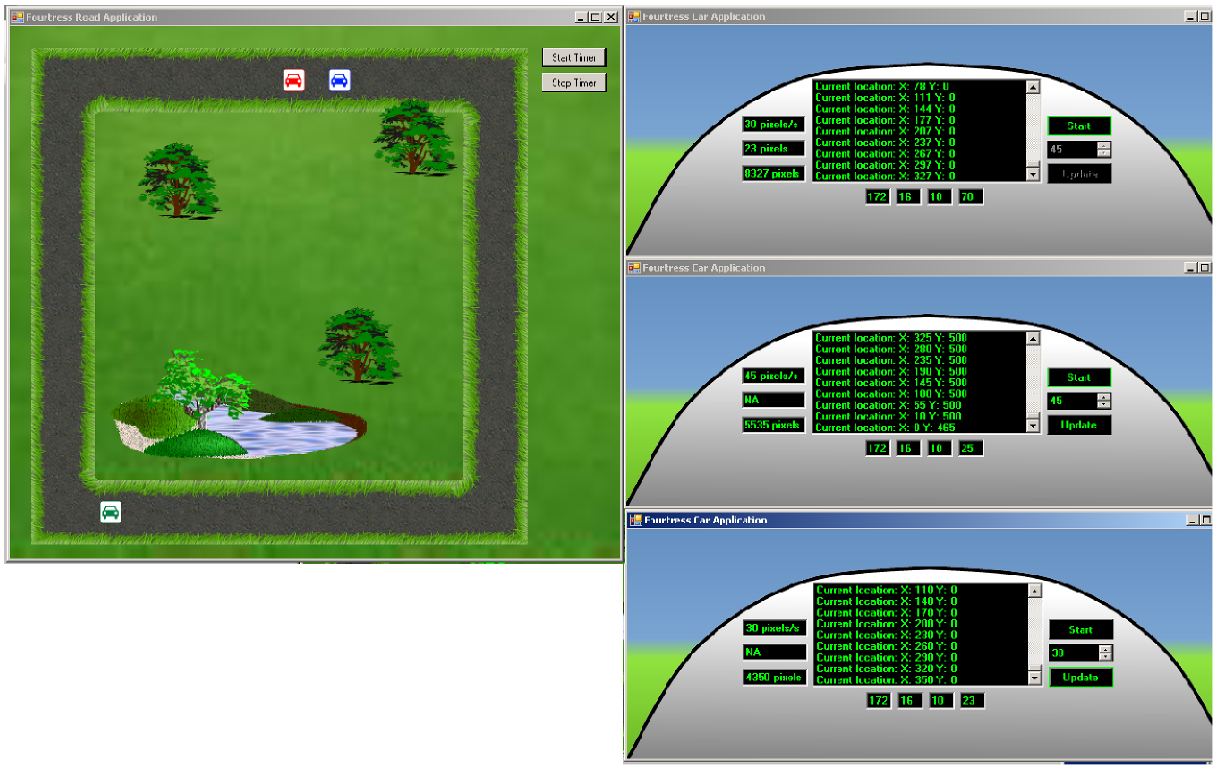
Figure 20: Road GUI

This phase finished one objective which was the creation of a monitor program for all the cars. In the end this objective was even more important than expected because it gives a good overview of the impact of the C-ACC algorithm.

7.5 Demonstrator

The final phase of the graduation period was spent on the demonstrator application. The demonstrator application makes it possible to show what the technology is capable of. It combines the road and car functionality to one software application. This application can instantiate multiple cars that are directly shown on the road. Although the main reason for Fourtress is to show what the Connect & Drive technology is capable off, it is also a good test environment for further progress in the overall Connect & Drive project.

The demonstrator completes one more objective, made at the start of the assignment. This objective was to create a good test environment in which everything is combined. Figure 21 shows the GUI of the demonstrator being used with a couple of cars instantiated.



```
Current location: X: 78 Y: 0
Current location: X: 111 Y: 0
Current location: X: 144 Y: 0
Current location: X: 177 Y: 0
Current location: X: 207 Y: 0
Current location: X: 237 Y: 0
Current location: X: 267 Y: 0
Current location: X: 297 Y: 0
Current location: X: 327 Y: 0
```

```
Current location: X: 325 Y: 500
Current location: X: 280 Y: 500
Current location: X: 235 Y: 500
Current location: X: 190 Y: 500
Current location: X: 145 Y: 500
Current location: X: 100 Y: 500
Current location: X: 55 Y: 500
Current location: X: 10 Y: 500
Current location: X: 0 Y: 465
```

```
Current location: X: 110 Y: 0
Current location: X: 140 Y: 0
Current location: X: 170 Y: 0
Current location: X: 200 Y: 0
Current location: X: 230 Y: 0
Current location: X: 260 Y: 0
Current location: X: 290 Y: 0
Current location: X: 320 Y: 0
Current location: X: 350 Y: 0
```

Figure 21: Demonstrator GUI

8 Conclusion and recommendations

At the start of the graduation period Fourtress set up a final goal for the assignment. That final goal was creating a software application that was able to control the speed of a actual car by using network communication and a C-ACC algorithm. This was a very ambitious goal that could only be achieved if the hardware for the car would be ready during the project.

During the project the hardware was not ready but a lot has been done and finished. The network communication between cars can be implemented in a real car because the cars are able to communicate with each other over the network without user interaction.

Other functions that work in a simulated environment are the speed control, the positioning system and a basic C-ACC algorithm. Cars used for the simulation can be started on multiple computers in a network and can then be monitored by the road application. These two applications are also combined to make one system which can be used as a demonstrator to show what the technology can do in real life.

As said, the current application can be used as a demonstrator on conferences, meetings and presentations. This makes it possible for Fourtress to trigger more companies to invest or investigate in these technologies. This can make the process of the project go faster and thus make it possible to use the technology earlier.

When the hardware for the real car becomes available Fourtress can start expanding the basic implementations that are done for the simulated environment. With minor modifications the software, in combination with the hardware, should be able to take control of a real car. By using dynamic libraries this is made even easier. When this step is done some real testing can be done and the results of those tests will help Fourtress improve the software.

The most important part of improving the software can be done on the C-ACC algorithm. The reason for this is that a real implementation of a C-ACC algorithm needs to take many more factors into account. These factors range from the weight and length of a car to the surface the car is driving on.

Another point of improvement can be done on the communication between cars. The current communication setup is okay but can be better by optimizing the data being sent between cars. This can make the cars communicate more information with the same amount of data being sent.

During the graduation period it also became clear that these two improvements for the final solution require additional research. This additional research will be done by external companies also participating in the Connect & Drive project.

Evaluation

At the end of my graduation period I can sit back and have a good look on the past five months. They went by very quick and that means I enjoyed my time at Fourtress. In this evaluation I will describe my graduation period the way I've experienced it.

The overall thought about my graduation period is that it went very well. At the start I did not have a good view of what the assignment was exactly about, but in a short period of time it all came clear to me. I was going to be part of a bigger project which could hopefully change driving and the way of living in the future.

The first steps were not too difficult as most of the things needed to be done were about testing what could provide me with enough details on network communication to get me started with programming. After some research and a lot of information read, I could already start with programming.

The programming of the software also went well and although there were some difficult tasks, I learned a lot (for example using DLL's and threads). The difficult tasks made me research the different technologies more in detail to solve the problem I had. When the research still did not help out the other software engineers at Fourtress could sometimes help me out. Most of the time, this gave a different point of view over the situation and pointed me in another direction which could solve the problem.

Before the graduation period I was not sure what to do after my graduation, but during my graduation period it became clear to me that the best choice is to find a job and start working instead of getting a master degree. The main reasons for this are that I enjoyed my stay at Fourtress and working all week. Another reason for this choice is that I do not have to go to classes and lectures anymore.

Bibliography

- Erich Gamma & Craig Larman, *Design Patterns*, Pearson Education Limited, 2005.
- Bjarne Stroustrup, *The C++ Programming* Language Special Edition, Addison-Wesley Pub Co, 2000
- Kennisinstituut voor mobiliteitsbeleid, *Mobiliteitsbalans 2008,* Kennisinstituut voor mobiliteitsbeleid, 2008.
- HTAS, *Connect & Drive Projectplan*, HTAS, 2008.
- MSDN Library, Microsoft, http://msdn.microsoft.com/nl-nl/library/default(en-us).aspx.
- Winpcap, Cace Technologies, http://www.winpcap.org.
- Boost Libraries, Boost, http://www.boost.org.

Appendices

Appendix I: Plan of approach