5th Semester Report
Roskilde Business College
School of Computer Science
6th of August 2002 – 11th of November 2002

CAR TRACKING SYSTEM

by
The Jacks
Dario Pacino
Hjörtur Sheving

The Jacks

Car Tracking System

"Programavimas šiandien tai lenktynės tarp programų sistemų inžinierių ir gamtos:
vieni stengiasi kurti programinę įranga visiškiems neišmanėliams,
o gandras vis atneša naujų idiotų.

Kol kas gamta pirmauja."

Rich Cook

A 5th Semester Report

By:
The Jacks,
Dario Pacino
Hjörtur Scheving

Students at:
Roskilde Business College,
School of Computer Science
Bakkesvinget 67
4000 Roskilde
Denmark

Project Period:
6th of August – 11th of November 2002

Supervised by:
Michael Claudius

Synopsis:
The knowledge and experience we have acquired through the last four semesters at RBC are to be brought together here in this report.

The report is build upon the development of a demo version of the Car Tracking System, which was developed by the Jacks for Sidabrinis Tinklas.

It is our hope that this report will proof to be of great help for future developers of the system.

Roskilde, 11th of November 2002

Title:

Car Tracking System

Keywords:

Rational Unified Process, RUP®

eXtreme Programming, XP

Mobile solution

Summary:

This report tries to describe the process of developing a demo version of the Car Tracking System, from A to Z. It follows the developers, two 5th semester students at Roskilde Business College, though their process, explaining their choices and decisions along the way.

Authors:

_____ _____

Hjörtur Scheving Dario Pacino

Table of Contents

Preface

This report is a part of our main thesis for the International Datamatician course at Roskilde Business College (RBC), were we bring together the knowledge and experience we have acquired through the last four semesters, furthermore it will explore our ability, as international students, to adapt to an international real life working environment. The project this report describes was undertaken in Vilnius, Lithuania in co-operation with Sidabrinis Tinklas (ST), a local software company.

The report is written with IT professionals in mind, either teachers of Computer Science or future developers. The contents and writing style of the report therefore requires the readers to have a broad knowledge, especially of Rational Unified Process (RUP®), eXtreme Programming (XP) and Object Oriented Analysis and Design (OOA&D).

The structure of the report is rather straight forward, mainly based upon the structure of RUP®. The report is divided into four main parts. We start by describing the process of selecting the appropriated software methodology. The second part is divided into the sets of RUP®, describing the general ideas and architecture of the system, together with everything else related to the process including the conclusions of each of the sets.

The third part is the most product oriented one, divided into the three subsystems were we cover the Requirements, Analysis & Design, Implementation, Test Set of each the three systems in more depth. At the end of this part we have, a Deployment Set were we finalise the practical part of the project and describe the hand over of the product.

Finally, yet importantly, in our fourth part, we have an Epilogue were we evaluated the process and finally try to answer our problem definition. We use many abbreviations through out the report, therefore we include a special abbreviation table in Appendix A in order to ease the reading.

The reader should consider, as RUP®/XP is an iterative process so there might be shown decisions in a set that are not described until in later sets, e.g. some of the things stated in the Analysis & Design Set are not explained before in the Implementation Set. However, this is an exception and should not hinder the reader in getting an understanding of the system and the process through the report.

Introduction

Since this is, a very product oriented report then in order to ease the understanding of what is to follow in this report then we include here a little introduction to the system we developed. Lets start with the project proposal presented to us by ST:

> *"Project proposal: Car-tracking system*
>
> *The project goal is to develop a car-tracking system using GPS (Global Positioning System) and GPRS wireless data transfer technologies in combination with a geographical information system (GIS).*
>
> *A full setup of the system is as follows. A numbers of cars drive in a pre-defined area (e.g., city of Vilnius). Each car is equipped with a GPS/GPRS device, which delivers car's positions to a central computer using GPRS wireless data transfer technology. The central computer has a GIS system installed (e.g., Akis, see www.akis.mii.lt), which displays received positions. Some of the cars are additionally equipped with computers having the same GIS system, so that they can receive positions of other cars from the central computer.*
>
> *Other information that can be sent between car computers and the central computer includes:*
> - *positions and states of other selected objects*
> - *object-related information*
> - *messages*

The tasks of the project include analysis of the system and programming of system components such as:

- *transfer of positions to the central computer*
- *displaying objects in a GIS system*
- *transfer of positions from the central computer to car computers*
- *transfer of other information*

The expected result is a functioning demo system. Programming can be done using C++ or Java™."

Even if the tasks seemed to be pretty set at the start, we found out that more things were actually required, and that the structure of system was mainly up to us.

The Car Tracking System (CTS) is a system designed for companies that have to manage mobile employees and a large fleet of vehicles. The system will allow the users to keep track of their vehicles, by visualising their geographical position on a Geographical Information System (GIS). Moreover, the companies will be able to exchange messages with all the vehicles and the system will allow the users to run queries on particular databases.

The main aim of the project was though, to create the system so that it is flexible towards future implementations and database connections. By this we mean that the system must be able to connect to different kind of Database Management Systems (DBMS) with minimum, if any, implementation changes, and that its architecture must be so that it can be tailored to fit different customers.

During the process, we divided the system into three essential subsystems: the Query System, the Messaging System and the Tracking System:

Tracking system:

The idea is that a company with a fleet of vehicles equip each car with a GPS/GPRS device, which delivers the car's position to a central computer. The central computer is then able to display the car's position on an installed GIS system enabling greater coordination and overview of the fleet.

Messaging system:

Even a sophisticated system like the Tracking system does not help much on its own. Even if the company knows where the cars are then they need to be able to communicate with them in order to coordinate the fleet. Of course, there are several options here; use normal mobile phones or short wave radio but these solutions do not enable an easy way of keeping records of the communications. Therefore, each car would be equipped with a computer and we would utilise the GPRS connection we already have between the cars and the central computer, making message passing between the cars and the central computer possible and store all communications into a database.

Querying system:

Accessing information can often be a tedious task for the mobile worker, but having the connection to the central computer can provide a remote access to a central database. This would give the mobile worker the opportunity to make searches in a central database, accessing the needed information within seconds.

Acknowledgments

First of all, we would like to thank Vilnius University and Roskilde Business College for giving us the chance to do our main thesis in Lithuania.

Michael, thank you for your guidance through out the project, we know it was not an easy process. Special thanks for believing in us, and trusting us to represent RBC in Lithuania.

Saulius, thank you for all you assistance during our stay in Lithuania, thanks to you everything in relation to the university went smoothly.

To everyone at Sidabrinis Tinklas, thanks for having us around. It was a pleasure to get to know you guys and thanks for all the help.

A very special thanks goes to Giedrius, for being so patient with the two crazy persons we are. With out you we would not have reached as far as we did during this project. We own you big time.

To Audrius and Ignas, guys we would never have made it without you!! We would have been homeless for the entire stay. Thanks for showing us around and teaching us how to use the trolleybuses.

There is more to life than just work; to all the bartenders and waitresses at Brodvejus that made our weekends rock, we love you, and we'll be back!!

Never have we in such a short time made as many friends as we did in Lithuania. To all our friends we left in there, thank you for making this an unforgettable stay, we will never forget you, however we will never forgive you for letting us go. See you all soon!!!

Problem Definition

The project can be divided up into two parts, a practical and a theoretical part. The practical part is rather straight forward provided to us by Sidabrinis Tinklas (ST). ST is a Lithuanian software company, located in Vilnius but stretches it's business angles worldwide with customers located e.g. Finland, Sweden, and Estonia to name a few. Armed with about 25 capable employees they specialise in mobile solutions, both "shrink-wrapped" and customised software.

Even though the Car Tracking System is supposed to be a "shrink-wrapped" product then this project has a connection with a costumer of ST that is interested in a system like this and will be able to comment it.

There is no doubt that it costs a lot of money, both the needed hardware and the manpower, to develop a full system of this magnitude. Therefore, in fact we can say that the demo is the bait for that costumer, to allure them into investing in this system. So, this project and its outcome can have substantial financial benefits for ST.

There are a couple of questions that come to our mind, when we look at the practical part of the project.

- Is it possible to use a demo version of a system to attract costumers?
- Can a customised system also become a "shrink-wrapped" product?
- Can two datamatician students go abroad and work with a local company as a part of their main thesis?
- How well are we prepared to handle methods/tools/techniques/technologies outside the curriculum of our studies?

The goal of software development is to produce quality, on time and within budged software but it is not as easy as it sounds. There does not exist a "Silver Bullet" to reach that goal but it is not all lost because there is help to be found. But where and what to look for? We will start in the jungle, the methodology jungle! There are literally hundreds of software development methodologies out there, ready to guide us towards the goal, and there we will start and end our theoretical part of this project.

We will try to evaluate different development models and methodologies and out from that choose one or sculpture our own development methodology that we will follow through out the project. We will argue and present valid reasons for our choice. Our main focus will be on answering the following

- **Does the chosen methodology suit the scope and size of our project?**

However, under this broad question we have many others:

- How well does the methodology cover the project management of our project?
- How well does the methodology cover the documentation of our project?
- Does the methodology cover the whole lifecycle of our project?
- Is it necessary to tailor the chosen methodology to our project and how easy is it to do so?
- Does the methodology give the developed system the necessary support it needs to go into further development after the project finishes?
- How easy is it to follow the methodology?
- Is the support and tools provided with the methodology adequate?

There are no simple answers to these questions. However, in the Conclusion part of the Epilogue we try to base our answers to these questions upon our experience gather though out this project.

Methodology

1. Introduction

It is not always that the project manager or the development team can choose themselves which software development methodology to follow, as a matter of fact that happens rarely nowadays. "..this is less likely nowadays, the project manager may have the power to decide on the most appropriate development model to be used for the project" (James Cadle & Donald Yeates, 2001). However, for this project the development team has the full authority to choose whichever methodology to follow.

Sure, we have preferences and suggestions from our customer, who uses RUP® as a standard in their company, but never the less we have tried to take an objective look into the jungle of methodologies. In this chapter, we try to describe this process but it is not our intentions nor is it possible to discuss all available development models and software methodologies.

2. Choosing the Methodology

2.1 Development Models

Basically, there are only two development models[1]: a waterfall model and a spiral model. There exist many others but they are merely variants or refinements of these two, way too many for us to discuss here in this chapter.

- ***Waterfall model:***

  Breaks the system development into number of sequential stages, where each stage has to be completed before entering the next one. The outputs from one stage are used as inputs to the next. There is no iterative work between the stages but can be with in a stage.

- ***Spiral model:***

  Often referred to as the iterative model, works in a different way than the waterfall model. It carries out the same activities over a number of cycles in order to produce the wanted system, clarifying the requirements, issues and solutions on the way.

For us there was never a question, which way we wanted to go, we were going spiral!

The waterfall model is a good model to follow if, and only if, the requirements and the business environment are stable. This is due to the sequential structure of it; it requires that you are able to finish one stage before entering the other. But when do you know all requirements upfront? This is very unlikely in today's business environment where things change rapidly.

As vigorously, as the waterfall model rejects changes, the spiral model embraces them. As only parts of the system are developed at a time then changes in requirements come early and before the whole system has been developed. This model gives the chance of delivering a working version of the system in a short time, giving the customer the satisfaction of seeing some return on their investment.

Even with the limited experience we have in software development, we have seen and experienced that working after a spiral model suited us much better in situations like our project, as we knew very little about the requirements of the system and we saw great risks ahead of us with many new technologies that we had no experience with. Furthermore, we believe that this gives a more correct system as the requirements are discovered through out the system development not all upfront. Divide and concur!

[1] Project Management for Information Systems, 3rd Edition

2.2 Agile vs. Heavyweight Methodologies[1]

It is not enough to have a model to follow we need a methodology to follow, or create our own. However you can say that there is no need to reinvent the wheel, it is out there. There are literarily hundreds if not thousands of methodologies existing today. They can be divided into two main groups: agile and heavyweight methodologies.

- ***Agile methodologies:***

  As the name indicates they are flexible, where the user can modify them to suit their own needs. They aim to deliver working software frequently, where the working software, not documents, is the primary measurement of progress. They promote close working relationship between the business people and the developers.

- ***Heavyweight methodologies:***

  When talking about heavyweight methodologies then bureaucracy is never far away. They have set steps to follow and are not flexible. They focus on documenting everything very carefully, believing that it will make quality software.

Few years' back no one had heard of agile methodologies, or lightweight methodologies, as they are also known as, but today it is one of the keywords of software development. It is somewhere in between the so-called "cowboy coding", where you just code and do not follow any methodology, and the heavyweight methodologies where you document everything.

The main creators of today's agile methodologies came together and found out that even though their methodologies had some differences then they could agree on a common manifesto[1]:

We follow these principles:

Our highest priority is to satisfy the customer
through early and continuous delivery
of valuable software.

Welcome changing requirements, even late in
development. Agile processes harness change for
the customer's competitive advantage.

Deliver working software frequently, from a
couple of weeks to a couple of months, with a
preference to the shorter timescale.

Business people and developers must work
together daily throughout the project.

Build projects around motivated individuals.
Give them the environment and support they need,
and trust them to get the job done.

The most efficient and effective method of
conveying information to and within a development
team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development.
The sponsors, developers, and users should be able
to maintain a constant pace indefinitely.

[1] Methodology resources

> Continuous attention to technical excellence
> and good design enhances agility.
>
> Simplicity--the art of maximizing the amount
> of work not done--is essential.
>
> The best architectures, requirements, and designs
> emerge from self-organizing teams.
>
> At regular intervals, the team reflects on how
> to become more effective, then tunes and adjusts
> its behaviour accordingly.
>
> *Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas*

We are rather practical oriented developers; with little interest in droving our motivation in paperwork. We also believe strongly that everything said in the manifesto helps and guides us in producing a quality software and project. Therefore, we will focus on agile methodologies.

2.3 Our Choice

Now we have narrowed our search in the methodology jungle but still a bit lost, as there are many methodologies that belong to the agile methodology group. However, after a brief introduction to eXtreme Programming (XP) on our third semester, we were already fascinated by it and eager to try it.

However, the company we are working with, uses Rational Unified Process (RUP®), and there we became for the first time familiar with it. There have been some discussions in the software development community if RUP® can be categorised as an agile methodology. There lies the beauty of it, its flexibility. It can be a heavyweight methodology or an agile methodology; all depends on the configuration of it by its user.

Since our objectives of this project are to produce software for a software company, which is not going to use the system but build upon it, and to produce a quality academic report, we really need to document the process and the product specifications it in a proper way. Therefore, we concluded that XP does not suit for our project because of its lack of documentation. However, RUP® will be suitable to help us to fulfil these objectives.

We are though not willing to completely give up on XP, nor is there any need to. There are some key features in XP that we want to try to integrate with RUP®; pair programming, refactoring, continuous integration and collective ownership. Where these practices come exactly into the process, we discuss in the later on in the Environment Set. However, we are not sure at this stage if RUP® and XP can really work together, that we will have to find out the hard way.

3. Conclusion

It was a very interesting process, were we explored new territories. This was the first time that we had free hands in choosing a methodology to follow. We already had some preferences when we went into the project but things are easier said than done, as the saying goes, and we realised that what we initially intended to use could not work to our satisfaction unless combined with another methodology. One thing we learned is that a system developer has to be flexible and quick to adapt to new practices, because we still believe that different methodologies apply to different type of project. It is not the same do develop a system for a space shuttle or a homepage for a local football club.

Environment Set

1. Introduction

The Environment Set focuses on providing the software development organisation with the software development environment - both processes and tools - that will support the development team. It is necessary for us, as for every development group, to evaluate the chosen developments processes, in our case RUP® and XP, and modify them and the related artefacts as needed.

2. Development Case

2.1 Introduction

2.1.1 Purpose

This document presents the manner in which the RUP® and XP development methodologies are to be used for the Car Tracking System project by the Development group The Jacks.

It defines the process configuration for this project. In particular, it defines the Process Roles supported in the Project and which Artefacts to produce.

2.1.2 Scope

This development case applies to the Inception, Elaboration, Construction, and Transition phases of the Car Tracking System project.

2.1.3 Overview

The remainder of this document describes ways in which the RUP® and XP will be adapted for this project.

Section 2 contains an overview of the development process, including project management and quality assurance activities. Section 3 describes the iteration workflows for the four phases. Section 4 describes the core workflows of RUP®, not just their workflow but also their artefacts and which artefacts have been eliminated and not least why they have been eliminated.

2.2 Overview of the Development Case

This project will consist of a combined Inception and Elaboration phase, a four-iteration Construction phase, and a small Transition phase. Design and code reviews will take place at key iteration milestones, and project quality reviews will be conducted on the fly.

2.3 Phases

2.3.1 Inception & Elaboration

Define the Scope and Vision, and Preparing the Development Environment

We will work with the stakeholders of the system to be developed to define the vision and the scope of the project. This will be done by meetings and reviewing the project proposal. We will produce the Vision document and the Business Case as artefacts. An initial version of the project risks will also be developed at this point. Researching and model the desired development lifecycle to be follow during the project and out line it in the Development Case.

Outline and Clarify the Functionality that is to be Provided by the System.

We will have meetings to collect stakeholders' opinions on what the system should do. We will outline the overall architecture of the system at this point as a basis for subsequent design activities.

Define the Software Development Plan.

With input from the Vision and Business Case, we will look into the resource estimates, the environment needed, and success criteria. We will also update the Risk List to refer to the identified use cases and add new identified risks. We will develop the initial Software Development plan to fully map out the project phases.

2.3.2 Construction

Clarify and Finalise the Requirements of the System

We will produce the Software Requirements Specification (SRS) of the whole system and specific ones for each individual subsystem. The SRS for the whole system will be delivered to the user company for verification. All needed documents will be updated to reflect on the up-to-date requirements.

Complete Component Development and Testing Against the Defined Evaluation Criteria

The system will be developed iteratively and incrementally in order to produce a complete product that is ready to transition to the user. This will be done by completing the analysis, design, development and testing of all required functionality for each subsystem at the time and then gradually intergrading them with the main system.

2.3.3 Transition

Prepare the System and its Related Documents for Deployment

A Deployment Document will be produced and all document related to the product that can be handy for further development of the system are up-to-date. The system along with its source code and documents will be handed over to Sidabrinis Tinklas.

2.4 Core Workflows

2.4.1 Introduction

This Development Case covers the configurations for all nine core workflows in The Rational Unified Process: Business Modelling, Requirements, Analysis & Design, Implementation, Test, Deployment, Configuration & Change Management, Project Management, and Environment.

2.4.2 Core Workflow Configuration

The purpose of this section is to explain how the core workflow configuration works. This includes the purpose of the different tables and sections that describe each core workflow.

Section: "Workflow"

This section details any changes made to the structure of the workflow itself. Typical changes include the addition of activities to describe our ways of working.

Section: "Artefacts"

The section describes, in a table, how the artefact will be used. These artefacts are mostly based upon RUP® templates but additional artefacts are added to the table as needed.

Artefacts	Created / Revised			Review Details	Tools used	Templates
	Inc/Elab	Const	Trans			
<Artefact name>	X	X	X	<Review procedure>		

The 'X' in one or more of the phase cells, means that we plan to revision of that artefact in that particular phase. In the cell Review Details we refer often to the Academic Committee or the Quality Group, these terms are explained in *the Project Management Set, section 3.3.2.* Finally, the Templates cell displays if there is a template provided by the RUP® or not.

Section: "Notes on Artefacts"

The main purpose of this section is to provide a list of all major artefacts that we want to exclude in this Workflow and the motives behind excluding them.

2.4.3 Business Modelling

As the project proposal from Sidabrinis Tinklas does not include anything regarding the business aspect of this system, we will not follow the workflow of the Business Modelling or produce any of the associated artefacts. Per say we consider the Business Modelling outside the scope of the project.

2.4.4 Requirements

Workflow

Requirements are captured through the development of use-cases and the Vision Document. Use-cases define actors, describing how the actors interact with the system. During the development of use-cases, a set of non-use-case requirements will also be captured. In the end, all requirements will be gather in a Software Requirement Specification for each subsystem.

Artefacts

Artefacts	Created / Revised			Review Details	Tools Used	Templates
	Inc/Elab	Const	Trans			
Actor	X	X		None	Microsoft Word®	No
Software Requirements Specification	X	X	X	Formal by Academic Committee and Quality Group	Microsoft Word®/Rational Rose®	RUP®
Supplementary Specification	X	X	X	None	Microsoft Word®	No
Use Case Specification	X	X		Formal by Academic Committee	Microsoft Word®/Rational Rose®	RUP®
Vision	X	X	X	Formal by Academic Committee	Microsoft Word®/Rational Rose®	RUP®

Notes on the Artefacts

Artefacts	How to Use	Reason/Notes
Actor	Used	No special document is required as the actors are few and simple enough to capture them with the Use-Cases
Boundary Class	Not used	Simple enough architecture to keep an overview with the System Architecture Document
Glossary	Maybe	It will be used if found necessary, but the close working environment and the small size of the development group makes it unlikely to be of great value

Artefacts	How to Use	Reason/Notes
Requirements Attributes	Not used	Due to the close working environment and the small size of the development group makes it unlikely to be of great value
Requirements Management Plan	Not used	Due to the close working environment and the small size of the development group makes it unlikely to be of great value
Stakeholder Requests	Not used	The stakeholders request come from meeting we have with our contact person at Sidabrinis Tinklas and not needed to be put in to a separate document
Supplementary Specification	Used	A separate document will not be produced but the supplementary specifications will be captured in the Software Requirement Specification
Software Requirements Specification	Used	A SRS for the Car Tracking System will be produced but also a separate SRS for each of the three subsystems.
Use-Case Model	Not used	An overview of the system will be captured in the Vision and in the Software Architecture Document.
Use-Case Package	Not used	Due to the close working environment and the small size of the development group makes it unlikely to be of great value
Use-Case Storyboard	Not used	The needed info will be captured in the Use-Case Specification
User-Interface Prototype	Not used	Sketches of the interfaces will be enough and placed in the SRS

2.4.5 Analysis & Design

Workflow

The use-cases developed during the Requirements workflow form the basis for subsequent analysis and design. Object-oriented design and analysis techniques will be used to complete the use-cases initially developed, produce the analysis and design object models and the software architecture document.

This is not a real-time system, so the real-time design workflow is omitted.

Artefact

Artefacts	Created / Revised			Review Details	Tools Used	Templates
	Inc/Elab	Const	Trans			
Analysis Model	X	X	X	None	Rational Rose®	No
Data Model	X	X	X	None	Rational Rose®	RUP®
Deployment Model	X	X	X	None	Rational Rose®	No
Design Class	X	X	X	None	Rational Rose®	No
Design Model	X	X	X	None	Rational Rose®	No
Design Package	X	X	X	None	Rational Rose®	No

Artefacts	Created / Revised			Review Details	Tools Used	Templates
	Inc/Elab	Const	Trans			
Event	X	X	X	None	Microsoft Word®/Rational Rose®	No
Interface	X	X		None	Microsoft Access®/Word®	No
Software Architecture Document	X	X	X	Formal by Academic Committee and Quality Group	Microsoft Word®/Rational Rose®	RUP®

Notes on the Artefacts

Artefacts	How to Use	Reason/Notes
Analysis Class	Not used	Due to the simplicity of the system we will sketch up the proposed classes on paper
Analysis Model	Used	This will be included in the Software Architecture Document
Capsule	Not used	Not a real-time system
Deployment Model	Used	This will be included in the Software Architecture Document
Design Class	Used	This will be done in the Design Model
Design Model	Used	This will be included in the Software Architecture Document
Design Package	Used	This will be included in the Software Architecture Document
Design Subsystem	Not used	We will split the systems up and produce separate designs for each system then we don't need this or you can say we already thought of this
Event	Used	This will be included in the Software Architecture Document
Interface	Used	This will be included in the Software Architecture Document
Protocol	Not used	We do not is the Capsule nor is it a real-time system
Reference Architecture	Not used	We don't have any reference architecture to rely on
Signal	Not used	Not of importance in our project
Software Architecture Document	Used	Quality Criteria will not be covered here because it will be covered in the Vision document. Process view will be included in the implementation view.
Use-Case Realisation	Not used	Not needed because of the simplicity of the use cases

2.4.6 Implementation

Workflow

Implementation will occur by developing objects and packages based on the design models developed earlier. Once these are initially created, the development team will review them, unit tested by the developers, and integrated into subsystems and the Car Tracking system for integration testing. Here we will add to the traditional RUP® some of the implementation practices of XP: pair programming, refactoring, collective ownership and continuous integration.

Artefacts

Artefacts	Created / Revised			Review Details	Tools Used	Templates
	Inc/Elab	Const	Trans			
Build		X	X	Formal by Quality Group	TextPad®, Java™ 2 SDK 1.3, Java™ VM	No
Component		X	X	None	TextPad®, Java™ 2 SDK 1.3, Java™ VM	No
Implementation Model		X		Formal by Academic Committee	Microsoft Word®/Rational Rose®	No

Notes on the Artefacts

Artefacts	How to Use	Reason/Notes
Build	Used	There will be no special documents associated with the Build. It will consist of our working demo of the Car Tracking System, and will grow as we produce and intergraded the subsystems.
Implementation Subsystem	Not used	We will produce an Implementation Model for the Car Tracking System and for each individual subsystem. Therefore, we have decided that there is no need for this the Implementation Subsystem.
Integration Build Plan	Not Used	The iterations are small and each subsystem is rather small at this demo stage, therefore, we have decided just to plan the Integration Build Plan on the fly and will not be documented. In addition, the development team only consists of two members that will be doing pair programming so there is no need to coordinate the activities of many programming pairs.

2.4.7 Testing

Workflow

Informal test cases and scripts that are developed from the use-cases will drive the test workflow. Using again some of the core practices of XP, (e.g. pair programming, collective ownership and continuous integration.), with the core Testing workflows described by the RUP®.

Artefacts

Artefacts	Created / Revised			Review Details	Tools Used	Templates
	Inc/Elab	Const	Trans			
Test Class		X		None	TextPad®, Java™ 2 SDK 1.3, Java™ VM, JUnit™	No

Environment Set

Artefacts	Created / Revised			Review Details	Tools Used	Templates
	Inc/Elab	Const	Trans			
Test Components		X		None	TextPad®, Java™ 2 SDK 1.3, Java™ VM, JUnit™	No
Test Evaluation Summary		X		None	Microsoft Word®	No
Test Plan		X		None	Microsoft Word®	RUP®
Test Results		X		None	Microsoft Word®	RUP®
Test Document		X		Formal by Academic Committee	Microsoft Word®	No
Test Script		X		None	TextPad®	No

Notes on the Artefacts

Artefacts	How to Use	Reason/Notes
Test Case	Not used	It has been decided that because this is only a demo version then we will not produce any Test Cases but testing is though done but only discussed between the developers and then implemented straight away, and the source code the only documentation of the test case.
Test Model	Not used	Because this is only a demo version and due to the close working environment we have decided that this artefact is not necessary for the development of the system.
Test Evaluation Summary	Used	Included in the Test Document
Test Plan	Used	Included in the Test Document
Test Document	Used	A test document will be created for each subsystem, covering what will be tested and then a short evaluation on how the testing went
Test Package	Not used	Because this is only a demo version and due to the close working environment we have decided that this artefact is not necessary for the development of the system.
Test Procedure	Not used	Included in the Test Document
Test Subsystem	Not used	Will be covered by producing a Test Document for each subsystem
Workload Analysis Document	Not used	Because this is only a demo version and due to the close working environment we have decided that this artefact is not necessary for the development of the system.

2.4.8 Deployment

Workflow

We will use the main RUP® deployment workflow but crammed into one artefact. We will hand in the product, Car Tracking System Demo version, and a simple user guide that will include the installation procedures. No formal training will be provided.

Artefacts

Artefacts	Created / Revised			Review Details	Tools Used	Templates
	Inc/Elab	Const	Trans			
Bill of Materials			X	None	Microsoft Word®	No
Deployment Document			X	Formal by Academic Committee and Quality Group	Microsoft Word®	No
Deployment Plan			X	None	Microsoft Word®	No
Deployment Unit			X	None	None	No
End-User Support Material			X	None	Microsoft Word®	No
Installation Artefacts			X	None	Microsoft Word®	No
Product			X	None	Java™ 2 SDK 1.3, Java™ VM	No
Release Notes			X	None	Microsoft Word®	No

Notes on the Artefacts

Artefacts	How to Use	Reason/Notes
Bill of Materials	Used	Because this is only a demo version and is only to be handed over for further development. Then we will cram this into the Deployment Document along with other Artefacts of this Set.
Deployment Document	Used	This is not a part of RUP® but we will use inspirations from the templates of the Deployment Set and build our own document.
Deployment Plan	Used	Because this is only a demo version and is only to be handed over for further development. Then we will cram this into the Deployment Document along with other Artefacts of this workflow.
End-User Support Material	Used	Will only consist of shot description of the system and included in the Deployment Document.
Installation Artefacts	Used	Because this is only a demo version and is only to be handed over for further development. Then we will cram this into the Deployment Document along with other Artefacts of this workflow.
Product Artwork	Not used	This demo is not to be marketed or any trademark to be registered so this is not needed.
Release Notes	Used	Because this is only a demo version and is only to be handed over for further development. Then we will cram this into the Deployment Document along with other Artefacts of this workflow.
Training Materials	Not used	As this demo is not to be used for anything else than further development and for supervised demonstrations then there is no need for a Training Materials

2.4.9 *Configuration & Change Management*

Workflow

For our project we have a lightweight configuration management process, we have put everything that belongs to Configuration & Change Management into Project Management. These issues will be cover in the Software Development Plan.

2.4.10 *Project Management*

Workflow

The project management part of the project will rely mostly on the produced plans: Software Development plan and the Iteration Plans. Those documents will guide us through the process and give us criteria to evaluate our process as well as guiding us with the Configuration Management of the project. Risks will also be managed during this continues workflow using the Risk List to try to mitigate us through them.

Artefacts

Artefacts	Created / Revised			Review Details	Tools Used	Templates
	Inc/Elab	Const	Trans			
Business Case	X	X		Formal by Academic Committee and Quality Group	Microsoft Word®	RUP®
Effort Sheets	X	X	X	None	Microsoft Word®	No
Iteration Assessments	X	X	X	None	Microsoft Word®	RUP®
Iteration Plan	X	X		Formal by Academic Committee and Quality Group	Microsoft Word®/ Microsoft Project	RUP®
Review Record	X	X	X	None	Microsoft Word®	No
Risk List	X	X	X	Formal by Academic Committee and Quality Group	Microsoft Word®/ Microsoft Project	RUP®
Software Development Plan	X	X	X	Formal by Academic Committee and Quality Group	Microsoft Word®	RUP®

Notes on the Artefacts

Artefacts	How to Use	Reason/Notes
Business Case	Used	Is only to give a brief overview and will not be kept up to date unless decided otherwise by the development group
Effort Sheets	Used	Are updated daily and kept to be able to evaluate if planed efforts measure against actual efforts
Measurement Plan	Not used	The metrics are simple in this project and are described in Software Development Plan

Artefacts	How to Use	Reason/Notes
Problem Resolution Plan	Not used	Due to the close working environment and small development group the problems will be dealt with on the fly
Product Acceptance Plan	Not used	The matter will be covered briefly in the Software Development Plan, and is of little importance here as this is a study project and no promises made of the out come of the project
Project Measurements	Not used	The metrics are simple in this project and are described in Software Development Plan and no special storage is needed for the outcome of them other than in the Academic Report
Quality Assurance Plan	Not used	The quality is mainly the concern of the Academic Committee and the Quality Group therefore the quality assurance lies with them but sure quality will also be considered by the development group and is addressed by meetings and reviews made by prior mentioned parties.
Risk Management Plan	Not Used	The risks are managed through the Risk List
Review Record	Used	Will be kept as meeting notes and as emails
Status Assessment	Not used	Status will be monitored in the meetings and through Iteration Assessments
Work Order	Not used	Due to the close working environment and small development group the work order is discussed on the fly

2.4.11 Environment

Workflow

The environment workflow focuses on the activities necessary to configure the process for a project. It describes the activities required to develop the guidelines in support of a project.

Artefacts

Artefacts	Created / Revised			Review Details	Tools Used	Templates
	Inc/Elab	Const	Trans			
Development Case	X	X	X	Formal by Academic Committee	Microsoft Word®	RUP®

Notes on the Artefacts

Artefacts	How to Use	Reason/Notes
Business Modelling Guidelines	Not used	The business aspect of this project is not important and is handled lightly by only producing a minimum Business Case
Design Guidelines	Not used	Due to the size and scope of the project we decided to keep the Design Guidelines built into the Software Development Plan
Development Infrastructure	Not used	This is handled in the Development Case, in a simple efficient way for a project of this size
Development-Organisation Assessment	Not used	The development group is fixed and external people cannot be added therefore it is pointless to make any effort in assessing the group

Artefacts	How to Use	Reason/Notes
Manual Styleguide	Not used	As a demo version is only to be developed at this stage and the produced user guide will be very limited therefore we will set the style of it in a informal way when we produce it
Methods, Tools, and Techniques	Not used	Due to the size of the development group and the close working environment we think this is covered in a satisfactory manner in the Software Development Plan and the Development Case.
Project-Specific Templates	Not used	We will mostly us RUP® templates and those styles will be kept through out all produced documents including the Academic Report.
Programming Guidelines	Maybe	The development group consists of two persons, Pair Programming will be conducted so the standards will just be set as we go and should not produce any problems. If needed for further development then these standards can be listed at the end of the project
Test Guidelines	Maybe	See Notes for Programming Guidelines
Tools	Not used	Covered in the Development Case
Tool Guidelines	Not used	Will be included in the Development Case if found necessary
Use-Case Modelling Guidelines	Not used	Again due to the size and close working environment we will just set this up along the way in an informal way
User-Interface Guidelines	Not used	This is only a demo version so user interface is not of great importance, we will only produce some sketches and try to get them approved by the user

3. Conclusion

RUP® is a big framework with a lot of activities and processes connected to it, there for it was essential for us to go through the whole RUP® process and evaluate it and try to decide what was necessary for us. It was pretty hard as we have never worked with it before, and we had different requirements than a normal software development project due to the academic side of it. Then we had to find ways to integrate the chosen XP practices to the whole process, and ways to document them. This was not done all in the beginning of the project; we started by building a frame to work from and then refined the working process along the way.

We only produced one document in this set, *Development Case*[1], even though RUP® has many more suggestions. This was done due to the size of the project and the development team and because of the close working environment. However, the prior mentioned document was of great use for use and the RUP® template was easy to follow covering the whole development process.

[1] Page 26

Project Management Set

1. Introduction

Project Management Set covers the art of balancing competing objectives, managing risk, and overcoming constraints to successfully deliver a product, which meets the needs of both customers and the users. The fact that so few projects are unarguably successful is comment enough on the difficulty of the task.

The chapter is divided into Risk List, where we try to analyse the risks associated with the project, Software Development Plan, where we have a general plan to follow, Iteration Plans, were we dig deeper into the planning of each iteration, and finally we have the Iteration Assessments, where we evaluate the success of the iterations.

2. Risk List

2.1 Introduction

This Document provides an overview of the risks connected to this project, describing them and the eventual actions to be taken.

2.1.1 Purpose

The risk list document has been designed to be ready to take actions in case problems arise during the project. It describes the risks associated with the project, including their priority probability and solution to them.

2.1.2 Scope

This Risk List Document goes beyond the scope of the Car Tracking System project for the customer; it also includes risks associated with the academic part of this project.

2.1.3 Overview

The Risk List is organised into two tables where the risks are listed in a priority order. For each risk, there is a description, a probability level, the impact that it will have, and mitigation and a contingency plan. The first table lists all the academic risks, which are defined with a higher priority in a study project. The second table lists all the risks related to the practical project.

2.2 Risks

2.2.1 Academic Risk List

ID	Description	Probability	Impact	Mitigation Plan	Contingency Plan
AR1	One of the two group member is not able to continue the project	1	*Level: 4* Lost of 50% of team members, and 50 % of workload increase on the other team member.	None	Contact supervisor. Resize project and scope. Re-plan project. Ask for time extension.

ID	Description	Probability	Impact	Mitigation Plan	Contingency Plan
AR2	One of the two group members becomes ill during the project and is unable to work at all his capabilities	2	*Level: 3* Delays on products delivery, and increase of workload for other members.	None	Lower ill member workload. Resize project scope, increase working hour if possible to other team member. Contact Supervisor. Ask for time extension.
AR3	Team members have personal problem that impact the team work	2	*Level: 4* Poor project quality	None	Contact Supervisor. Consider friendly negotiation or splitting the project into two.
AR4	Project cannot finish on due date.	3	*Level: 4* Poor project quality, non-terminated report.	Careful planning of each of the iterations, with project scope resizing if needed.	Contact Supervisor. Drastically stop the practical project and concentrate on academic report, emphasising on the reason of project failure. Or Resize scope to present achieved situation, and finish academic report.

Follow-up

The table below is a follow-up of the Academic Risk List; it displays how the risks are positioned in relation to probability and impact. The risks positioned in the first quarter of the table are the most relevant for our project.

PROBABILITY						
IMPACT		5	4	3	2	1
	5					
	4			AR4	AR3	AR1
	3				AR2	
	2					
	1					

From the Follow-up table it can be concluded that, regarding the academic part of this project, no real threads are present, beside the delivery due date.

2.2.2 *Practical Risk List*

ID	Description	Probability	Impact	Mitigation Plan	Contingency Plan
PR1	Hardware devices do not work or in expected manners	2	*Level: 3* Long delays if other devices are not available at that time. Impossibility to continue the project.	Do not use the devices in an improper way and make sure to acquire sufficient knowledge before using.	Stop working on the devices and transform the demo software in to a simulation that does not need the devices.
PR2	The User company is very slow at sending feed back.	4	*Level: 3* Delays on the project and days with very low productivity.	Appear active to the company and demonstrate that work is being done.	Take own decision and explain to the User Company that due to restricted time some decision has been taken, but that they can be changed.
PR3	The User company does not accept the decisions made if the previous risk had happened.	3	*Level: 3* Delays on project realisation, and waste of work.	Take as flexible decision as possible. Try to push the company a little to receive faster responses.	Explain to the company that the decision will be changed, but they will stay as they are for the demo version of the product.
PR4	Insufficient knowledge of tools to complete the project	3	*Level: 3* Delays on the project and low quality of end product	Choose best practices methods and tools, but prioritising known ones if possible	Resize the scope of the project, consult with supervisors

Follow-up

The table below is a follow-up of the Practical Risk List; it displays how the risks are positioned in relation to probability and impact. The risks positioned in the first quarter of the table are the most relevant for our project.

		PROBABILITY				
		5	4	3	2	1
IMPACT	5					
	4					
	3		PR2	PR3/ PR4	PR1	
	2					
	1					

From the follow-up tables we can conclude that the major risks go around the User Company. The lack of a fast communication could cause serious delays. The choice of tools and practices should be made with particular attention, as the learning risk "PR4" seems to be of big importance for the success of the project.

3. Software Development Plan

3.1 Introduction

3.1.1 Purpose

The objective of this Software Development Plan is to define the development activities in terms of the phases and iterations required for implementing the Car Tracking System and to produce the academic report for the 5th semester theses at Roskilde Business College.

3.1.2 Scope

This Software Development Plan describes the overall plan to be used by the development group The Jacks for developing the Car Tracking System and the academic report. The details of the individual iterations will be described in the Iteration Plans. The plans as outlined in this document are based upon the product requirements as defined in the Vision Document.

3.1.3 Overview

This Software Development Plan contains the following information:

Project Overview – provides a description of the project's purpose, scope and objectives. It also defines the deliverables that the project is expected to deliver.

Project Organisation – describes the organisational structure of the project team.

Management Process – explains the estimated schedule, defines the major phases and milestones for the project, and describes how the project will be monitored.

Technical Process Plans – provides an overview of the software development process, including methods, tools and techniques to be followed.

Supporting Process Plans – this includes the configuration management plan.

3.2 Project Overview

3.2.1 Project Purpose, Scope, and Objectives

Primarily objective of this project is to demonstrate the capabilities of the developers to applying the acquired knowledge from the four previous semesters at Roskilde Business Collage in practice. On the other hand, the scope of this project is to build a demo version of the Car Tracking System, in order to be able to practically demonstrate the potentials of the product. The product is being designed to facilitate and speed up the process of information research, and vehicle fleet coordination for a variety of companies.

3.2.2 Assumptions and Constraints

- During the development of the demo of the Car Tracking System, security will not be considered as this is only a demo version and showing functionality is of more importance to the user than having a secure system.
- The project has a deadline, the 11th of November, set by Roskilde Business College.
- As this is an academic project then it is limited to the two members of the development group, The Jacks.

3.2.3 Project Deliverables

Artefact	Receiver	Date of delivery
Academic Report, including everything below	Roskilde Business College/ Sidabrinis Tinklas	11/11/02
Software Requirements Specification	Sidabrinis Tinklas	18/10/02
Demo version of the Car Tracking System	Sidabrinis Tinklas	18/10/02
Other product related documents	Sidabrinis Tinklas	18/10/02
Source Code	Sidabrinis Tinklas	18/10/02

3.2.4 Evolution of the Software Development Plan

The Software development plan will be updated every time a new iteration starts. Artefacts and releases list will be updated as well as the phase plan.

3.3 Project Organisation

3.3.1 Organisational Structure

The Team is composed of only two members:

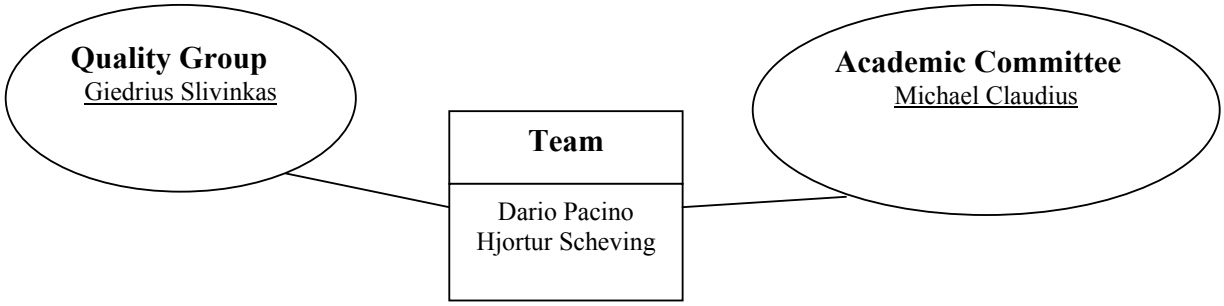
Dario Pacino, 5th semester student at Roskilde Business College

Hjörtur Scheving, 5th semester student at Roskilde Business College

3.3.2 External Interfaces

The Team is connected to two external groups: Academic Committee and Quality Group.

The Academic Committee is composed of Michael Claudius the official project supervisor. The Quality Group is composed of only one member Giedrius Slivinskas. The Academic Committee checks the quality of the academic report, while the Quality Group supervises the quality of the practical products.



3.3.3 Roles and Responsibilities

Due to the small project group and the close working environment, then the roles and responsibilities have not been divided between the two members. Both members are equally responsible of all tasks needed for developing the Car Tracking System and the Academic Report. However, the roles and responsibilities that apply to both members are those defined by the Rational Unified Process®[1].

3.4 Management Process

3.4.1 Project Plan

Phase Plan

The development of the Car Tracking System will be conducted using a phased approach where multiple iterations can occur within a phase. Inception Phase and the Elaboration Phase have been combined into one phase due to the small size and short timeframe of the project. For the same reasons, Iteration 1 overlaps both the Elaboration Phase and the Construction Phase. The phases and the relative timeline are shown in the table below:

Phase	Iterations No.	Start	End
Inception/Elaboration Phase	0, (1)	06.08.2002	22.08.2002
Construction Phase	1, 2, 3, 4	19.08.2002	14.10.2002
Transition Phase	5	15.10.2002	18.10.2002

The phase plan does not apply to the Academic Report.

Iteration Objectives

Phase	Iteration No.	Description	Associated Milestones
Inception/ Elaboration	0	Analyse system and prepare Project Environment.	- M1-Establishment - M2- Architectural Design - M3-Query System Proposal

[1] RUP® Framework, support documents and tools

Phase	Iteration No.	Description	Associated Milestones
Elaboration/ Construction	1	Develop a flexible interface that can connect to different types of databases implying little changes to the interface. Write Software Requirements Specifications for the general system and also specially for the Query and the Messaging system	- M4-Software Architecture Document - M5- Iteration 2 Plan - M6- Software Architecture Document - M7- Test Units - M8- DB Interface Unit - M9- Software Requirements Specification
Construction	2	Develop a flexible querying GUI that can adapt itself to the database in use. And integrate it with the database interface to combine them into the complete demo version of the Querying System	- M10- Iteration 3 Plan - M11- Test Units - M12- Query System - M13- All Documents Updated
Construction	3	Develop the Messaging System	- M14- Iteration 4 Plan - M15- Software Architecture Document - M16- Test Units - M17- Messaging System - M18- Query/Messaging System
Construction	4	Develop the Tracking System	- M19- Software Architecture Document - M20- Test Units - M21- Tracking System - M22- The Car Tracking System - M23- Iteration 4 Plan
Transition	5	Finalise systems integration an deliver the demo	- M24- Deployment Document - M25- Product Related Documents - M26- The Car Tracking System

Releases

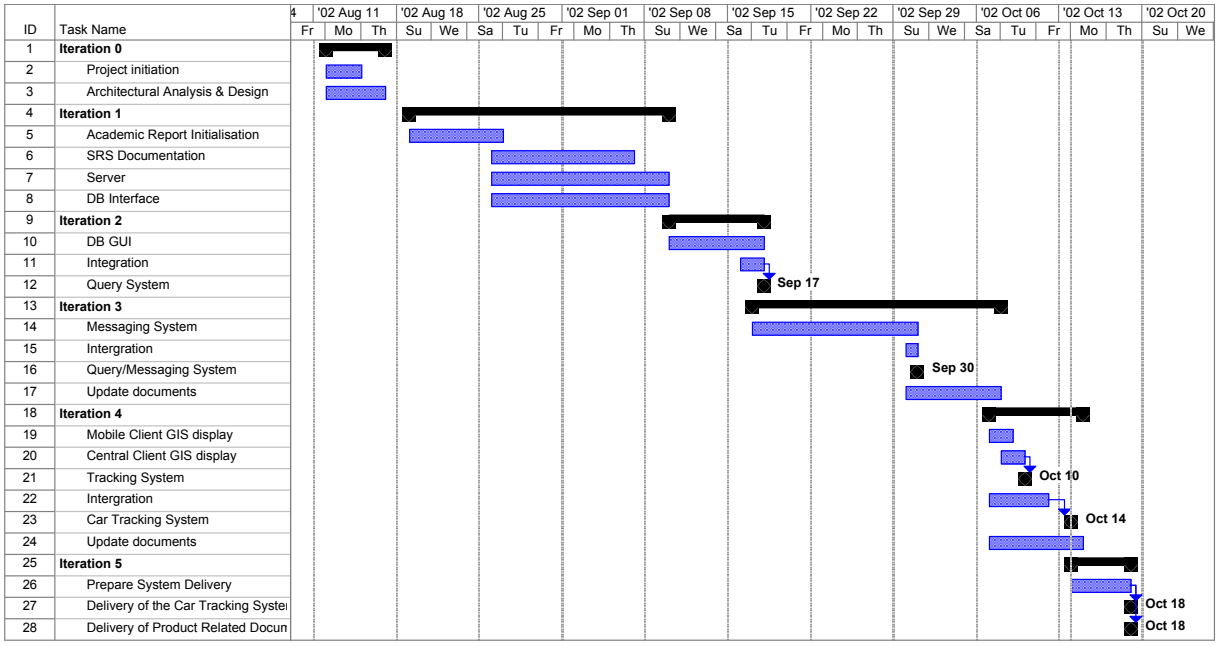
Key features as defined in the Vision Document are targets for four releases.

Release	Type	Functionality	Date of delivery
Querying System	Demo	The mobile user is able to send queries to the database and get results displayed on the screen. He/she can then look through 20 latest results. The central operator is able to do the same and pass the result to the mobile users if needed. The central operator can also see all connected mobile users	17/09/2002

Release	Type	Functionality	Date of delivery
Messaging System	Demo	The mobile user and the central operator are able to communicate with each other through the system, seeing all messages sent and received in the current log in session. However, the central operator can communicate with all mobile users but the mobile users can only communicate with the central operator. The mobile user can send a specific Action Result to the central operator relating to a certain result from the database, this will be displayed on the screen of the central operator. No information is stored in this demo version	30/09/2002
Tracking System	Demo	The Mobile User can see his/her own location on the AKIS system. The Central Operator can see the location of all Mobile Users on the AKIS system on his/her screen. However, because of absent of the GPS device we will work only with computer generated data regarding the locations of the vehicles.	14/10/2002
Car Tracking System	Demo	The Car Tracking System with the user interfaces, both for the central operator and the mobile user is ready, and communicates as needed with the AKIS system. Combining all the functions of the Query, Messaging and Tracking system.	14/10/2002

Project Schedule

A Gantt chart is included to give a better overview over the development of the Car Tracking System:



We only included the initial stage of the academic report as that would be the most consuming part of it otherwise we decided neither to include the academic report in the above Gantt chart or make a separate Gantt chart for it. The academic report is created in parallel to the whole software lifecycle, as its main substance is the RUP® documents that we make along the way. Therefore, we saw no reason to plan it specifically.

3.4.2 Project Monitoring and Control

Schedule Control Plan

The schedule will be controlled using effort-monitoring timesheets[1] and by monitoring if, set milestones are being met. Delays in the schedule will be compensated with additional working hours if applicable. In case the project goes out of schedule, the plan will be reviewed. If delays continue then the scope of the iteration will be changed (see Risk List, Risks: PR1, PR5, AR4)

Quality Control Plan

Practical product quality will be ensured by the use of testing tool throughout the process. Software releases will go through the Quality Group, which will have to accept the software before it can be released.

Academic products are sent to the Academic Committee that will revise them and send feedbacks. From the feedback, correction will be made. There are two possible scenarios, either the feedback is sent through email or through a meeting. It the case of an email then it will be stored. In the case of a meeting then a meeting, report will be produced covering the evaluation.

Reporting Plan

The size and type of this project does not require any paper report for the practical part of the project. The Quality Group is situated in the same office and reporting meetings are arranged on the fly, or at the reaching of each milestone. However, a meeting report[2] is produced after each meeting covering the agenda and noting down decision and future steps of the development.

To the Academic Committee, reports are sent as needed, presenting deliverable artefacts that have been created or gone through major modifications.

Measurement Plan

Effort and time will be used to track progress of the project. Planned vs. actual Effort-monitoring timesheets will be used by the development group to measure progress. Nevertheless, our main tool for measurement are our milestones that are easy to evaluate if reached or not through their evaluation criteria.

Close-out Plan

At the end of the project a conclusion will be drawn, considering the effectiveness of the process as a whole, lessons learned and included in the Academic Report.

3.5 Supporting Process Plans

3.5.1 Configuration Management Plan

Document Storing Procedure:

Every document name must be formatted as follow: <name>_<version >. The revision sheet on top of the document must be filled out every time a document is changed. Old copies of the documents must not be deleted until the final document is released. The document files are kept on a server at Sidabrinis Tinklas and backed up by their internal procedure.

[1] Appendix K

[2] Appendix L

Source Code Storing Procedure:

A folder has to be created for each workshop. Microsoft Visual SourceSafe® 6.0 is used to keep track of our source files. The source files are kept on Sidabrinis Tinklas database, the program creates a new version every time some modifications have been made and enable us access on every version of the code we make. It also makes sure that the two developers do not work on the same file at the same time with a check out and in procedures on each file, locking them for editing unless checked out.

Academic Report Storing Procedure:

Same as Document Storing Procedure but the reports will be stored in a different folder.

4. Iteration Plans

See Appendix B.

5. Iteration Assessments

5.1 Introduction

5.1.1 Purpose

The objective of the Iteration Assessments is to capture the result of the iterations, the degree to which the evaluation criteria were met, and a lesson learned and changes to be done

5.1.2 Scope

These Iteration Assessments applies to all the iterations of the Car Tracking System.

The successes of the iterations are measured against the milestones and their evaluation criteria as outlined in the Iteration Plans[1] for the system.

5.1.3 Overview

This document evaluates the iterations, summarising the objectives that were met, how well each of the iterations ran according to its plan.

Each of the iterations has its own section where the results are evaluated according to the criteria established in the Iteration Plan for each milestone.

Finally, the section lists the external changes that affected the iteration, such as changes in requirements, and problem areas that require future rework.

5.2 Iteration 0

5.2.1 Iteration Objectives Reached

The project was established and the environment set for the project. A common general vision of the system has been established but still there are some unclear requirements left. An overall design of the system has been made and agreed system architecture of the Query system has been made.

5.2.2 Adherence to Plan

The iteration executed according to plan completing on schedule.

[1] Appendix B

5.2.3 *Results Relative to Evaluation Criteria*

The following table lists the milestones evaluation criteria as outlined in the Iteration Plans and comments on how well the criteria for success were met.

Milestone	Evaluation Criteria	Iteration Results
M1	Agreement that the right sets of requirements have been captured and that there is a shared understanding of these requirements.	The general requirements were understood but some specific requirements were still a little unclear.
M1	Agreement that the schedule estimates, priorities and development process is appropriate.	Was not completely accomplished. The Development Case document is not ready though the development process has been decided among the developers. No future dates have been set for these documents but should be finished as soon as possible. The schedules have been approved. Nevertheless, with better understanding of the system these are bound to change.
M1	All known risks, at the time, have been identified and a mitigation strategy exists for each, where applying.	The risks have been identified and a mitigation strategy has been made for each of them.
M2	The System Architecture Design document should contain the proposed solution for the architectural structure of the query system, emphasising on how long it will take to add a database, and the time for implementation.	A document listing three options that we have for the Query system, listing their advantages and disadvantages. Moreover, evaluated on their criteria.
M3	The Architectural design should include the basic architecture of the system. The document should include a UML model of the system, and list all the possible architectures that could be used, the one proposed and a brief explanation of the reason it has been chosen	A decision was made to make a document listing the two “easier” options for the Query system. The document was then delivered to the user company

5.2.4 *External Changes Occurred*

None

5.2.5 *Rework Required*

The Schedules and the Risk list have to been updated as needed.

5.3 Iteration 1

5.3.1 *Iteration Objectives Reached*

Clarifying some of the systems requirements was a big part of the Iteration 1 and was necessary in order to produce an up to date SRS for the system. The Query System has been analysed and designed down to implementation level. The Query interface has been created and is now only missing the GUI part of the system.

5.3.2 *Adherence to Plan*

Very early on, a week was added to the original plan due to changes in the requirements of the system. However, it did not hurt, as a week shorter was needed for the Query Interface. In addition, 2 days were added to allow the developers to establish their Academic Report part of the project. Moreover, at the end of the iteration, we added the weekend as working days as we were behind on the schedule; therefore, we also moved the some of the milestones from Friday to Monday.

5.3.3 *Results Relative to Evaluation Criteria*

The following table lists the milestones evaluation criteria as outlined in the Iteration Plans and comments on how well the criteria for success were met.

Milestone	Evaluation Criteria	Iteration Results
M4	The SAD should include all the use case view and logical view of the system, in the form of UML diagrams.	All known use cases were documented.
M5	The Iteration Plans Document should be updated and should include a detailed plan for iteration 2.	Has been accomplished. Iteration 2 plan is now included and the whole document been updated.
M6	The SAD should include use case, logical and implementation view of the Database interface part of the Query System	A special SAD for the Query System has been done capturing the use cases, logical and implementation views.
M7	A list of test classes made using JUnit™ is ready to be used during testing phases	This Milestone was unrealistic, because the test cases were implemented on the fly, a small code, a small test or vice versa. Therefore, the test classes were not ready until the end of the iteration.
M8	Java™ Implemented Units should be ready and well tested. JUnit™ tests should give a 100% correctness result, and so should the integration test.	The Units were ready and all working correctly. With 100% result from the tests.
M9	The SRS documents should be approved by our Quality Group	This was accomplished and documents send to user pending evaluation.

5.3.4 *External Changes Occurred*

The User changed and clarified some it's requirements causing extra workload and changes of the original Iteration Plan.

5.3.5 *Rework Required*

All schedules and documents produced so far for the system need to been updated with latest requirements.

5.4 Iteration 2

5.4.1 *Iteration Objectives Reached*

A user interface has been created both for the mobile user and the central operator and integrated with code produced in Iteration 1.

5.4.2 Adherence to Plan

A day was added to the original plan due to some unexpected complications with the use of layout manager in Java™. Because of this the milestones were moved a day.

5.4.3 Results Relative to Evaluation Criteria

The following table lists the milestones evaluation criteria as outlined in the Iteration Plans[1] and comments on how well the criteria for success were met.

Milestone	Evaluation Criteria	Iteration Results
M10	The Iteration Plans Document should be updated and should include a detailed plan for iteration 3.	Has been accomplished. Iteration 3 plan is now included and the whole document been updated.
M11	Test classes for every unit of the Query System, that can be tested with JUnit™ are ready.	Has been accomplished, but it is not possible to test the GUI itself using JUnit™ so there were not many new test classes created. Giving 100% result
M12	The system has to meet the criteria set in the *Iteration Plans, section 1.4.4*[1] and approved by the Quality Committee.	The system meets the criteria, 100% and has been approved by the Quality Committee but there are some minor defects and some refactoring left to do. Some days have been dedicated for this purpose in Iteration 3 Plan.
M13	A consistency is kept through out the project documents.	This Milestone was not completely met, but will be reached within the 19th of September.

5.4.4 External Changes Occurred

None

5.4.5 Rework Required

All schedules and documents produced so far for the system need to been updated with latest requirements. Furthermore, some minor defects and some refactoring on the Query System are left to do. Implementation and Testing documents for the Query System have to be made.

5.5 Iteration 3

5.5.1 Iteration Objectives Reached

A user interface has been created both for the mobile user and the central operator supporting the messaging part of the Car Tracking System and integrated with code produced in Iteration 1&2.

5.5.2 Adherence to Plan

The development of the Messaging System itself went according to plan if not even a little better, however a week was added to the original plan, as we needed to update many of our documents. Because of this the milestones M14 and M15 were moved a week.

5.5.3 Results Relative to Evaluation Criteria

The following table lists the milestones evaluation criteria as outlined in the Iteration Plans and comments on how well the criteria for success were met.

[1] Appendix B

Milestone	Evaluation Criteria	Iteration Results
M14	The Iteration Plans Document should be updated and should include a detailed plan for Iteration 4	Has been accomplished. Iteration 4 plan is now included and the whole document been updated
M15	The Software Architecture Document should include use case, data model, logical and implementation view of the Messaging System.	Was accomplished and has been intergraded into the academic report
M16	Test classes for every unit of the Messaging System, that can be tested with JUnit™ are ready	Has been accomplished, but it is not possible to test the GUI itself using JUnit™ so there were not many new test classes created. Giving 100% result
M17	The system has to meet the criteria set in *Iteration Plans, section 1.5.4*[1] and approved by the Quality Committee	The system meets the criteria, 100% and has been approved by the Quality Committee
M18	An integrated version of the Query and the Messaging system with a common main GUI.	Was accomplished and approved by the Quality Committee

5.5.4 External Changes Occurred

None

5.5.5 Rework Required

None

5.6 Iteration 4

5.6.1 Iteration Objectives Reached

A connection between the Car Tracking System and a GIS has been made and the GIS can show the position of the mobile users, both locally on the mobile users computer and on the central operators computer.

5.6.2 Adherence to Plan

Everything went according to plan.

5.6.3 Results Relative to Evaluation Criteria

The following table lists the milestones evaluation criteria as outlined in the Iteration Plans[1] and comments on how well the criteria for success were met.

Milestone	Evaluation Criteria	Iteration Results
M19	The Software Architecture Document should include use case, data model, logical and implementation view of the Tracking System.	The document was ready on time, covering all prior mentioned sections
M20	Test classes for every unit of the Tracking System, that can be tested with JUnit™ are ready.	Has been accomplished, but it is not possible to test the GUI itself using JUnit™ so there were not many new test classes created. Giving 100% result

[1] Appendix B

Milestone	Evaluation Criteria	Iteration Results
M21	The system has to meet the criteria set in *Iteration Plans, section* 1.6.4[1] Plan and approved by the Quality Committee.	The system meets the criteria, 100% and has been approved by the Quality Committee
M22	An integrated version of the Car Tracking System with a main GUI and running AKIS[2] parallel to it in order to display the positions of the cars.	Has been accomplished.
M23	The Iteration Plans Document should be updated and should include a detailed plan for iteration 5.	Iteration 5 plan is now included and the whole document been updated.

5.6.4 External Changes Occurred

None

5.6.5 Rework Required

A meeting is set with the User Company on the 16th of October, therefore some new requirements might come up and some rework on the Software Requirements Specifications might be necessary.

5.7 Iteration 5

5.7.1 Iteration Objectives Reached

The deployment of the Car Tracking System was successful.

5.7.2 Adherence to Plan

Everything went according to plan.

5.7.3 Results Relative to Evaluation Criteria

The following table lists the milestones evaluation criteria as outlined in the Iteration Plans and comments on how well the criteria for success were met.

Milestone	Evaluation Criteria	Iteration Results
M24	The Deployment Document should include a list of documents and files that are part of the deployment, an installation guide, guide for launching the applications for the Car Tracking System. It should also list all known errors and problematic features of the system.	The document was produced covering the set criteria.
M25	All products related documents should be finalised and up-to-date ready to be deployed to Sidabrinis Tinklas.	All documents ready and handed over to Sidabrinis Tinklas.
M26	A final version of the Car Tracking System from the Jacks with Java™ comments for all the classes.	Has been accomplished and the system handed over to Sidabrinis Tinklas.

[1] Appendix B

[2] Akis © V.Paliulionis

5.7.4 External Changes Occurred

None

5.7.5 Rework Required

None

6. Conclusion

Now we have tried to manage projects both using the waterfall model and now with the spiral model. And we are convinced that it is easier to successfully manage a project after a spiral model, especially if the developers themselves are also the project managers, unless you have a very stable business environment and software requirements. Project management is mainly covering two things time planning and risk management. And by the iterative approach, you uncover new risks quickly and you learn from your previous iterations, therefore you can quickly fix your time schedule according to lessons learned.

The RUP® documents we produced supporting the project management process, were of great help. Risk List[1] was produced in the start of the project and was under continuous revision through out the project. This focused us on avoiding risks when ever possible. The Software Development Plan[2] gave us a good overview over the whole project and was revisited after each of five iterations and updated, this gave Sidabrinis Tinklas good overview over the project and what and when to expect certain things. Furthermore, the Iteration Plans[3] deliver a more detailed plan into the iterations, which were assessed using the Iteration Assessments[4] that helped us to look back and learn from our experience. However, we found that the Effort Sheets[5], that we initially planed to use, were of no help to us because we did not plan everything in such details that we could compare them with our plans. Instead, we found that the milestones set in the Iteration Plans[3], were a better evaluation tool to monitor our progress.

[1] Page 38

[2] Page 41

[3] Appendix B

[4] Page 47

[5] Appendix K

Requirements Set

1. Introduction

The Requirements Set covers the artefacts that help in managing the systematic approach to finding, documenting, organising and tracking the changing requirements of a system. This is essential in order to create a common understanding of the system between the developers and the users.

First the chapter presents a Business Case, where we try to address the business aspect of the system, then we go to the Vision document where we try to establish a common view of the system between the developers and the business people, and last but not least a Software Requirements Specification is presented, covering the requirements of the system as a whole.

2. Business Case

2.1 Introduction

The Business Case document provides an overview of the project and gives a brief reason why this product is of business interest.

2.2 Product Description

The product is a car tracking system suitable for companies that need to keep track of their vehicle fleet and allowing their mobile workers to access a central database remotely. The system should run on a mobile computer situated in a car, and should be able to connect to a central computer. The server in the central computer will have a database of information that can be queried. The information will be exchanged between the central system and the mobile unit, via the Internet using the new GPRS wireless connection technology. In addition, every vehicle will be equipped with a GPS device. Connecting the central computer and the client to a GIS, the two parties can receive information about the position of the cars within the specific area of the GIS, with the possibility of visualising special information such as car-status or messages.

2.3 Business Context

This project is aimed at create a commercial product that is suitable for a variety of companies that need to keep track of their vehicle fleet. At this initial state, the product only is developed up to a demo version that can demonstrate the real product potentialities.

2.4 Product Objectives

The product objective is to ease the processes of information research, and improve the coordination of the vehicle fleet. The use of mobile computers in the cars will enable the mobile worker to do searches from their cars. Thanks to the GPRS technology, it will be possible to maintain a constant on-line connection to a central computer, with out requiring any bigger devices than a mobile phone. Using the GPS, it will be possible to send and display the position of the cars in the GIS, which will give a better overview of the situation and it will ease and speed up the coordination of the vehicle fleet. To improve communication, the messaging service will allow the users to send messages between the cars and the central system. It will also allow the central system to broadcast messages to all the cars in one go.

2.5 Constraints

- The Project will be running concurrently with an academic project as well.
- Programming language for the system: C++ or Java™
- Time frame: 6/8/2002 – 11/11/2002
- Hardware:
    - GPS/GPRS device
    - GPRS enable mobile phone

3. Vision Document

3.1 Introduction

3.1.1 Purpose

The purpose of this section is to collect, analyse, and define high-level needs and features of the Car Tracking System. It focuses on the capabilities needed by the stakeholders and the target users, and why these needs exist. The details of how the Car Tracking System fulfils these needs are detailed in the Use-Case[1] and in the Software Requirement Specification[2].

3.1.2 Scope

The scope of the Vision is to give an overview of the system as a whole, and to characterise the system features. It will change if the system requirements or features will change.

3.2 Positioning

3.2.1 Business Opportunity

The software product produced will be targeted for a wide market place. The flexibility of the system will allow the product to be sold and configured to different companies, and with a minimum amount of installation and configuration work.

3.2.2 Problem Statement

The problem of	Managing a fleet of vehicles and communicating with mobile workers and keeping logs on those communications. Furthermore, not having a direct connection to a central database for the mobile workers
Affects	The mobile workers, their customers and the company as a whole
The impact of which is	A lack of coordination among the mobile workers and a delay in information research. Furthermore, miscommunications can occur and no way of verifying the communication, that went on.
A successful solution would	Centralise the coordination of the vehicles, keep logs of all communications and would give the mobile workers the access to the needed information.

[1] Appendix C

[2] Page 64

3.2.3 *Product Position Statement*

For	A variety of companies that need to manage a fleet of vehicles and mobile workers
Who	Need to gain fast access to information and effectively coordinate action.
The Car Tracking System	Is a tool
That	Enables on-line database query, message exchange, keeps a log of those messages and can display the current position of the vehicles.
Unlike	Old radio communication systems.
Our product	Provides a mobile communication and connection to database and a GIS system.

3.3 Stakeholder and User Descriptions

3.3.1 *Stakeholder Summary*

Name	Represents	Role
Giedrius Slivinskas	Customer company & contact person to user company	On-Site Customer

3.3.2 *User Summary*

Name	Description	Stakeholder
Mobile Worker	Uses the system as a mean to get valuable information	Giedrius Slivinskas
Central Operator	Uses the system as a mean to coordinate the vehicle fleet and the mobile workers	Giedrius Slivinskas

3.3.3 *User Environment*

There are two types of users for this system, mobile users and a central operator. The mobile user will be using a vehicle as his/her workspace. The number of vehicles connected to the central computer can vary depending on the user company that will embrace the system. Being the systems environment, the vehicle, the physical space for hardware is limited. The central operator can either be situated in a vehicle or more likely at a central office managing the coordination's of the vehicles.

3.3.4 *Stakeholder Profiles*

Giedrius Slivinskas

Type	Database expert, Project Manager at Sidabrinis Tinklas.
Responsibilities	Contact person to the user company, and ensuring that needed hardware and support is provided to the developers by Sidabrinis Tinklas. Moreover, he evaluates any releases of the system and documents related to it.

Success Criteria	The development of the system is not delayed because of lack of communication to the user company or due to lack of needed hardware or other support provided by Sidabrinis Tinklas. His evaluations reflect on the requirements of the user company.
Involvement	Quality coordinator and mediator with user company and Sidabrinis Tinklas.

3.3.5 Key Stakeholder or User Needs Related

Need	Priority	Concerns	Current Solution	Proposed Solutions
Query System	High	The mobile users don't have a access to the central database	Radio communication is used where the mobile users asks a central operator to access the central database and then the info is related through the radio system	User would like to have online access to the database from the car using a GPRS enabled mobile phone and being able to look through previous query results. Furthermore, allowing the mobile users to send Action Results related to the query result.
Messaging System	Medium	No records are kept on the communication between the mobile users and the central operator.	All communications go through the radio system.	User would like to be able to send and receive messages between the car and the central computer. Still using the same GPRS connection. Moreover, keeping logs with all communications between the two parties.
Tracking System	Medium	Managing and coordinating a large fleet of vehicles is hard without any oversight over the positions of the vehicles.	The central operator has no way of knowing the position of the vehicles unless the mobile users radio in their position.	The vehicles will be equipped with GPS devices that send the positions of the vehicles to the central computer, which will display them using the integrated GIS of the Car Tracking System.

3.4 Product Overview

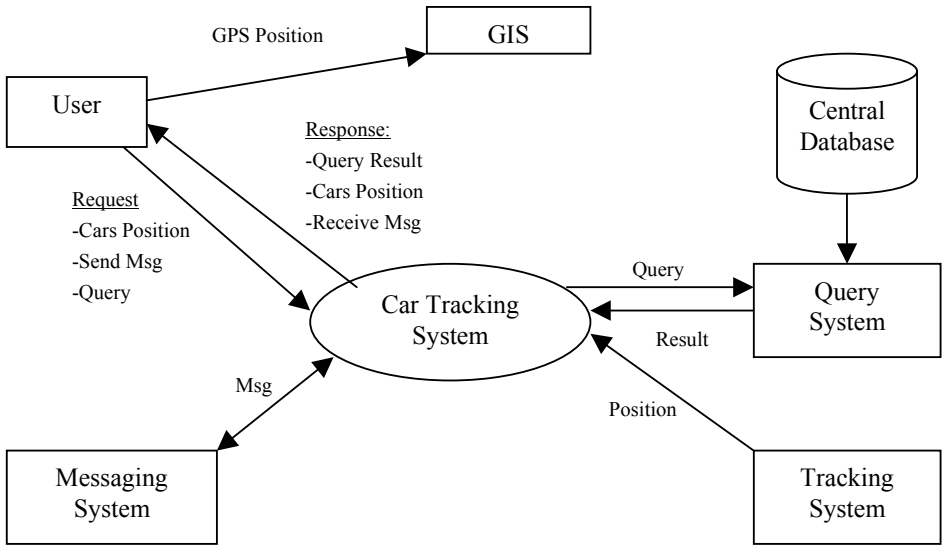
3.4.1 Product Perspective

The Car Tracking System will be a completely new system designed to ease the process of gathering information from a central database and managing a fleet of vehicles. It will consist of a client component for the central database, and two client-server components for the messaging and the location information.

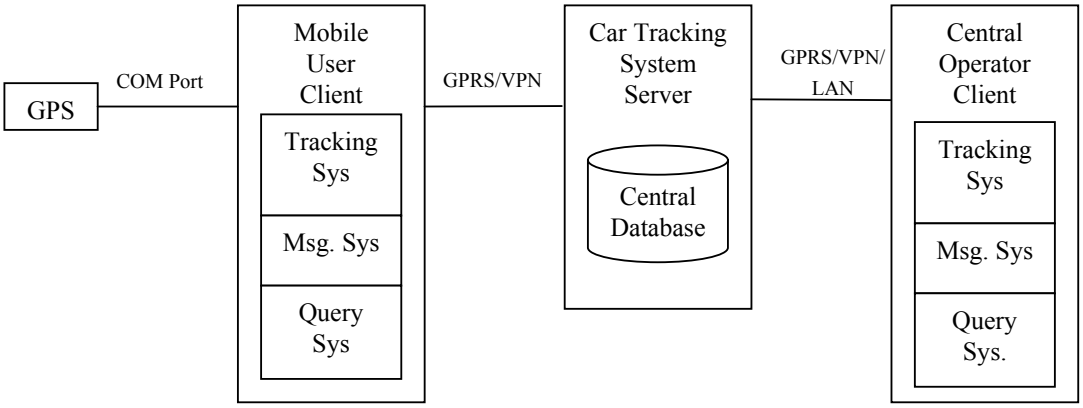
The system will run on two clients, one for the mobile users, placed in the vehicles, and one for the central operator for the administrative and coordination operations. Both Clients communicate to and through a central server, which is used as an interface to one or more database servers, and as a mean of communication between the clients.

The main idea of the system is to have a flexible architecture that can adapt itself to different databases with the minimum changes to the system.

System Context Diagram



System Overview



3.4.2 Summary of Capabilities

The following table describes the main capability of the Car Tracking System, in terms of benefits and supporting features.

Customer Benefit	Supporting Features
Ability to see the position of the vehicles at any time, allowing easier coordination of the fleet.	On-line Tracking System, based on GPS devices and GPRS wireless technology, displaying the results using an integrated GIS.
Faster research of information.	On-line database connection, accessible from the vehicles at any given time.
Fast reporting of actions taken upon related topics.	With every query result, it is possible to send to the central operator a message, which tells what actions have been taken regarding a certain result.
Possibility of message interchange between vehicles and central operator, which will help the coordination.	On-line messaging accessible from the vehicles and the central operator at any time.
Ability to see transcripts of the communications between the mobile users and the central operator.	A database that keeps a log of all the communications.
No need for software change if changes are made to the database	Self-modelling interfaces that read the structure of the database from a provided text file, from the user built on a protocol set by the developers.
24-hour on-line connection to the system.	GPRS technology allows the system to have a permanent connection to the internet/LAN/VPN.

3.4.3 Assumptions and Dependencies

The following list describes all the assumption and dependencies related to the system.

- Customer Company provides the team with GPS and GPRS devices, for all the implementation stages of the project that utilise those technologies.
- The suggested GIS, Akis[1], can and permission granted to integrate it with the Car Tracking System.

3.5 Product Features

3.5.1 Logon

All users of the system must logon and provide sufficient authentication of them.

3.5.2 Search on Databases

The users are able to make searches in the connect database, using different types of criteria, depending on the part of the database that is to be addressed or the kind of information to be retrieved. The results are then displayed on the screen of the requesting party.

3.5.3 Send/Receive Messages

The Mobile Users are able to send and receive messages to and from the Central Operator; the messages that have been sent/received during the login session, of the users, are displayed on the screens of the Mobile Users.

However, the Central Operator can choose to which Mobile user he/she wants to send a message to, or even broadcast a message to all the Mobile Users.

3.5.4 Send/Receive Query Result Without Request

The Central Operator can send a specific query result to a particular Mobile User or even broadcast it to all Mobile Users. The Message then pops up on the Mobile Users screen.

[1] Akis © V.Paliulionis

3.5.5 Monitor Cars

The system will keep track of the current positions of the vehicles, using the integrated GIS system to display them and the user will be able to monitor the vehicles movements. The cars will be displayed on the map of the GIS as car icons images together with the name identifier.

3.5.6 Send Action Results

With every query result, the Mobile Users gets, they can choose from a list or enter some Action Results related to that specific result and send it to the Central Operator, and then furthermore it will be saved on a database.

3.5.7 Add/Delete/Modify Users Information

The system will provide the facility to add, delete or modify the users information.

3.5.8 Add or Delete Database Structures

The system will provide the facility to add or delete Database structures, thereby controlling which databases are connected to the system at a given time.

3.5.9 View Past Query Results

The users are able to select to view past query results, navigating through them with the press of a back and forward button.

3.5.10 Save/View Communication Logs

All communications between the Mobile Users and the Central Operator will be saved on a database. Then the Central Operator can choose to view any prior communications from the main menu.

3.5.11 Send/Receive Objects

The Central Operator can send to the Mobile Users objects that need to appear on the integrated GIS, e.g. a building. The sent object then visualises on the screen of the Mobil Users, showing the position of that object.

3.5.12 View Online Users

All online users are displayed on the screen of the Central Operator, allowing him/her to keep track of all users.

3.6 Constraints

- The systems have to use GPS device and GPRS wireless technology
- Programming must be done using Java™ or C++

3.7 Quality Ranges

Availability	24 hours – 7 days a week.
Usability	The system should be easy to use. New users should be ready to use the system within few days.
Maintainability	No maintainability quality ranges have been set at this stage.
Comprehensibility	The system should be very comprehensible for the users, as they will have limited computer knowledge and training with the system. Making the system easy and logical to use is therefore considered quite important.

Efficiency	The efficiency of the system it is important because the system handles confidential data that should be transferred correctly.
Security	Security is an important issue for the final system. The confidential information that is transferred should not be intercepted and corRUP®ted by any intruders. The data should only be changed by authorised personnel outside and not in any connection with this system. However, those issues will not be implemented in this demo version.
Testable	Testing is an important part of any system, it will not be possible to ensure any efficiency if the system could not be tested.
Flexibility	Flexibility is one of the most important sides of this system. The system should be able to be configured to different kind DB servers with the least number of changes.
Reusability	As previously explained in the Flexibility Quality, this system is very likely to be reused and connect to other DB servers. Therefore, particular attention should be paid at design and implementation phases.
Portability	Portability seems not to be a very important part of this system, but as the system is to be designed to be flexible, some thought about portability could be made, especially at implementation time.

The system should be a highly reliable and an efficient system, based on a flexible architecture that allows changes with the least amount of work. It should also be possible to test the correctness of the system. Little thought into security and portability should also be done, but only as examples or suggestions at design time.

3.8 Precedence and Priority

The following list defines the priority with witch the system had to be built.

- Query System
- Messaging System
- Tracking System

3.9 Other Product Requirements

3.9.1 Applicable Standards

- The communication will relay on TCP/IP protocols.
- The whole system will be platform independent, in order to be as flexible as possible.
- The connection will use GPRS technology.
- The vehicles position shall be determined by the use of GPS technology.

3.9.2 System Requirements

- The system should interact with a database in the most flexible way. Any kind of DBMS should be able to connect to the interface and the GUI, causing the least number of changes in the software.
- The GUI should be user friendly and self-describing.
- The system should be continuously on-line, using GPRS technology for communication.

3.9.3 Performance Requirements

None at this stage.

3.9.4 Environmental Requirements

None at this stage.

4. Software Requirement Specification

4.1 Introduction

4.1.1 Purpose and Scope

The Software Requirement Specification describes all currently known functional and non-functional requirements of the Car Tracking System. This document helps keeping track of what the system actually has to perform and with what characteristic or constraints. It can also be used to as a mean to clarify the system functionality with the User Company.

4.1.2 Scope

This Software Requirement Specification refers to the requirements of the Car Tracking System as a full system, not only the demo version of it.

4.1.3 Overview

This document is organised by kind of requirements, going through the functional ones to the design constraints and supplementary specifications.

4.2 Functional Requirements

4.2.1 Main Application

Authentication

The level of access to the system is secured by a login procedure that allows only authorised access. The access level of the user determines which application is to be run, the Mobile-Client or the Central-Client.

4.2.2 Mobile-Client Application

Launched Automatically

The application should be launched automatically every time the computer is turned on and should be the only programme running on the computer.

Querying

Mobile User is able to select a database and run queries against that database using a GUI. The Querying criteria are specified in S*ection 4.6.6*. The query result will be displayed using a GUI, following the graphical criteria described in *Section 4.3.1*.

View Received Query Results Locations

If a Query Result has an addressed linked to it, then the Mobile User should be able to see the location of it, displaying it using a GIS.

Log all Query Transactions

Mobile User is able to see a log of all his/hers query transactions; both received and sent within his/her log in session (‘Query Sent’, ‘Result Received’). The log also contains the results received without a request from the Central Operator. Moreover, it should be possible to select a specific “Result Received” from the list and view it; the number of results stored depends though, as the number can be configured in each application (see *Section 4.6.9*).

Messages

Mobile User is able to send messages to and receive messages from the Central Operator. The system displays received messages following the graphical format described in *Section 4.3.1*. All sent and received messages from the last login time should be visible in a scrollable text area.

Action Results

For each Query Result received, the Mobile User is able to input a reply, an Action Result. The reply can either be selected from a list of choices or typed in by the user. Each database has a pre-defined list of replies, which can be configured by the Central Operator.

View Own Location

Mobile User should be able to see his/her location displaying them using a GIS.

Lifetime of Displayed Objects

The objects displayed on the GIS will be displayed until deleted by the Mobile User.

4.2.3 Central-Client Application

Listing Online Users

Central Operator is able to see all Mobile Users that are connected to the server application.

Querying

Central Operator is able to select a database and run queries against that database using a GUI. Querying criteria are specified in *Section 4.6.6*. The query result will be displayed using a GUI, following the graphical criteria described in *Section 4.3.1*.

View Received Query Results Locations

If a Query Result has an addressed linked to it, then the Central Operator should be able to see the location of it, displaying it using a GIS.

Log all Query Transactions

Central Operator is able to see a log of all his/hers query transactions; both received and sent within his/her log in session, (‘Query Sent’, ‘Result Received’). Moreover, it should be possible to select a specific “Result Received” from the list and view it; the number of results stored depends though, as the number can be configured in each application (see *Section 4.6.9*).

Sending Query Results Without Request

Central Operator is able to send a query result to selected Mobile Users, and if needed connect a message to it. Furthermore, the Central Operator can choose if the result should be displayed directly on the screen of the selected mobile client or if the mobile client can choose when to display it.

Messages

Central Operator is able to send a message to selected Mobile Users, if they are online. All sent and received messages from the last login time should be visible in a scrollable text area. The Central Operator should be able to set the importance of the message, normal or critical. If normal then display on only within the text area but if critical then a pop up window should also be used to display the message at the mobile client.

Action Results

Central Operator is able to view Action Results of the Mobile Users within the message text area.

Configuration of Action Results List

Central Operator is able to create a list of Action Results for a selected database, as well as modify and delete such lists.

Configuration of Databases

Central Operator is able to add, remove or modify a database, which can be used for querying. When adding a database, the Central Operator must specify a path for a text file describing the database structure (its format is given in *Section 4.6.7*). Once the Central Operator has added, removed or modified a database used for querying, the mobile client updates to reflect those changes. During this update, the mobile client is not available to the Mobile Users.

The server must validate the text file format, which is representing the database structure, and give a message to the central operator about the validation success.

User Management

Central Operator is able to add, modify and remove users of the system. He/she may also assign access permissions to different databases and functionalities.

View Mobile Clients Locations

Central Operator should be able to see the location of all the online Mobile Users displaying them using a GIS.

Display Locations on the Mobile Client GIS

Central Operator should be able to display objects locations on the screen of a selected Mobile User using a GIS to display it. Furthermore, he/she should be able to connect a message to the object.

4.3 Usability Requirements

4.3.1 Graphical User Interface

Graphical user interfaces of client applications should be as simple and intuitive as possible.

4.4 Reliability and Performance Requirements

None at this stage

4.5 Supportability Requirements

None at this stage

4.6 Supplementary Specifications

4.6.1 Display Connection Status

Show the connection status, between the Clients and the Server, at all time. This must be done in a clear manner and in a graphical way, e.g. red and green light indicating the status of the connection.

4.6.2 Display a Detailed Progress Bar

Display the progress of transactions, how much percentage is done of the transaction.

4.6.3 Display Acknowledgements

Acknowledgments should be displayed for all transactions. This should be done by a brief text on a progress bar; unless an error occurs in the transaction then a pop up message should be displayed reporting so.

4.6.4 Keyboard and Mouse

Limit all keyboard and mouse actions due to the nature of the suggested hardware. See *Section 4.8.2*.

4.6.5 Query Format

Queries from the server application to the DB Server will be sent using SQL.

4.6.6 Search Criteria

The client and server applications will accept the following search criteria for queries:

=	(searches for fields with values that are equal to the given value)
LIKE	(searches for fields with values that have matching patterns with the given value)

Given values for both criteria are case insensitive.

4.6.7 DB Structure Format

The server application accepts the following format text file for the DB structure:

```
<TABLE>
      NAME  table_name
<FIELD>
      NAME  field_name
      TYPE  field_type
      SEARCHABLE  Y/N
</FIELD>
.
.
</TABLE>
```

Example:

```
<TABLE>
      NAME  car
<FIELD>
      NAME  ID
      TYPE  TEXT
      SEARCHABLE Y
</FIELD>
<FIELD>
      NAME Picture
      TYPE  GRAPHIC
      SEARCHABLE N
</FIELD>
</TABLE>
```

This file would describe a database table named “car” that has two fields. The first field is named “ID” and is of type “TEXT”; queries can be run on this field. The second field is called “Picture” and is of type “GRAPHIC”, which means that it will contain an image; this field is not searchable.

Note:

TYPE Allowed	Description
TEXT	String format (500 characters allowed)
GRAPHIC	Image in JPEG or GIF format
INT	32 bit Integer Number
FLOAT	32 bit floating-point number
BOOLEAN	Y or N value. Y stands for true and N for false

4.6.8 Action Result Type

Valid Action Result type has to be in the format of a text string.

4.6.9 Configuration of the Number of Query Results Stored

Users are able to configure how many Query Results the application keeps for further viewing.

4.6.10 Log all Transactions

A log should be kept on a database about all transactions in the system, e.g. all messages send and received, all queries made, all Action Results etc. These logs should be viewable from the system by the Central Operator.

4.6.11 Configure Updates of Objects in GIS

It should be configurable, for the users, how often the location is updated of the displayed objects on the GIS.

4.7 Online User Documentation and Help System Requirements

None at this stage.

4.8 Interfaces

4.8.1 User Interfaces

Querying GUI

The interface needs to be able to model itself automatically depending on the database it is to be used for.

General GUI

The user interfaces should be as simple and intuitive as possible. The application should be in one window occupying the whole screen, with no pop up windows

Query Result

The query result has to display the exact time and data of when the specific result was received. Furthermore, it should also show the criteria that were searched to achieve this specific result.

4.8.2 Hardware Interfaces

GPS Device

The system has to be able to communicate with the GPS device.

GPRS Enable Device

Yet to be determined.

Keyboard

Limited keyboard with maybe only 15 keys.

Mouse

If needed then an easily operated mouse, as users may be wearing gloves while operating the application.

4.8.3 Software Interfaces

The system has to interact with a GIS system.

4.8.4 Communications Interfaces

The server application and the DB server will be directly connected via a LAN and will communicate using JDBC drivers and SQL.

4.9 Licensing Requirements

None at this stage.

4.10 Legal, Copyright, and Other Notices

None at this stage.

5. Conclusion

Like most other activities, this was not a one-time activity, but a continuous process through out the process.

One of the most crucial activities in the software process is to gather and manage the requirements of the proposed system. In our case, we started with making a Business Case[1], but soon found out that the business aspect was out of our scope for this project, as Sidabrinis Tinklas assigned us to the specific task of developing the system nothing else. Therefore, we left out all other business documents suggested by RUP®, and actually, we did not even bother of updating the Business Case[1] as it was of no importance for us.

The Vision document[2] though was a bridge between the developers and the business people; it gave us a common vision of the system. We believe it should help future developers, and others that come in any way towards this system, to get an overview over the system and the objectives of it.

However, it turned out that the Software Requirements Specification[3] was one of our most valuable document. It helped creating a detailed understanding of the system and the functionality of it one. It served as a very important link between Sidabrinis Tinklas and us as this was the document that we handed over to them after each major update of it, and in return got feedback from them on our understanding of the system.

[1] Page 56

[2] Page 57

[3] Page 64

Analysis & Design Set

1. Introduction

The Analysis & Design Set focuses on creating a robust architecture of the system. The artefacts for this set have been produced in order to help in transforming the requirements into a design of the system-to-be.

Differently then the traditional Object Oriented approach, we will not run two different sections for analysis and design, but we will combine them due to the size of the system. At first glance, we see that the classes and operations involved in this system are too few, in our opinion, to use time for detailed analysis. Furthermore, the division of the system in subsystems makes the product even easier to analyse.

In this chapter, we show the process that we followed in order to select the architecture in which the system is to run. First a little introduction into the way we worked on the Analysis and Design then we present a series of researches we made and present a proposal of different architectural solution. Further on we show witch architectural design has been choose, then end with the Software Architecture Document[1] of the general architecture of the system. More detailed architecture of each of the three subsystems can be found in appropriated subsystem chapter.

2. Our way

We will use some of the practices from analysis to get an overview of the system, but without spending to much time on details. We will produce logical views of the main classes with just enough information to have an idea of what the architecture of the system will be. To show the main functionalities we will refer to the Use-Cases specification[2] realised during the Inception phase. However, as we go down to a single subsystem, we will start the design of each of its components. Differently than the traditional Object Oriented approach, we will not design all the parts of a subsystem in one go, but we will go through one and start the implementation phase. Once that part has been developed and tested, we will start the design of the next part of the subsystem.

During the design, we will produce a detailed design model of the specific component using UML diagram that will include the main attributes and operations of each class. Together with the diagram a brief description of the classes will be given. If necessary, state chart diagram will be used for the respective classes. Furthermore than the diagram, a list of the main functions will be compiled and a specification of the most complex one will be created.

The documentation of these activities will be showed in the Software Architecture Document[1] of each subsystem, which will be updated as needed.

Due to the simplicity of the system, activities like class and event evaluation will be run on the fly and will not be documented. Eventual changes in the Analysis and Design of the system will be asserted during next iterations.

3. Class Description

3.1 Introduction

Hereafter are brief descriptions of the main classes of the system. Since an iterative approach is used during this development, then occasionally classes and parts of the system might be displayed in diagram but explained in later stages. Therefore this section has been created in order to aid in the understanding of the critical classes in the system and their functions.

[1] Page 72

[2] Appendix C

3.2 The Core Classes

Server Class

This class represents the server of the system. It manages all the system transactions and holds the system configuration.

MobileClient Class

The MobileClient class is the representation of the client application resided on the vehicles computers. It interacts with the system via the Server class.

CentralClient Class

This class is an extension of the MobileClient class, and represents the client application resided on the central computer. It has the same functionality as the MobileClient class with a few extra features, such as system configuration.

Result Class

The Result class holds the data received from the result of a query sent to the database.

ActionResult Class

For every Result, it is possible to send a response on what has been done concerning the task connected to the Result. The ActionResult class holds the information referring to this connection.

Message Class

All messages exchanged are passed trough the Message class, which also holds information such as sender and receivers of the message.

Database Class

This is the representation of the database structure to which the system is connected. All information such as tables, file path, users etc.

User Class

This class holds data about the users in the system, e.g. personal data, access level and privileges.

4. General System Architecture

4.1 Introduction

This documents presents the possible architectural solutions for the Car Tracking System, but with special emphasise on the Query System.

4.2 Purpose

This document has been created to give a better overview of the problem and its possible solution. It is created mainly for the user company, giving them the power of choosing a solution that suits them.

4.3 Type of Network

It is obvious, from the system requirements (*see page 64*), that the Query System is a distributed system.

The idea of sending query requests and receiving results gives us the idea of a System based on Client-Server architecture. Arguments could be set about the architecture looking at the messaging system. The Client-Server architecture does not seem to be so correct once we talk about exchanging messages between clients. Therefore, we decided to question the user Company in that matter and realised that messages are only sent between a Mobile Client and the Central Operator. Therefore, we decided to go on with the Client-Server architecture and use the Server as a dispatcher for the messages. If in future implementation the system will work with more than one Central Client, the Server could be used as a filter for the messages.

The user company has an already installed server with which our system will have to communicate with. The choice that was placed on us was if we had to implement our system through a LAN or through the Internet. In the tables below, we list the advantage and disadvantage of using the two technologies, in relation with the Car Tracking System.

	Internet
	Possibility to use the software from any computer online.
	New computers do not need installation. The browser will download the software when needed.
	Security measure will have to be considered more seriously. The possibility of intruders is higher.

	Local Area Network
	Security is important but the possibility of intruders in drastically reduced.
	Available only inside network. Opening gateway will decrease the security level.
	New computers will have to be configured to access the LAN.
	Lack of documentation about GPRS and LAN

Even after our research, we are still not sure about what to choose. The security issue could be critical to the User Company, so the decision will be participated to the next iteration. This choice will tough not cause any delays because the application can still be implemented. Implementing a commercial application means implementing an application that can suit more than one user. Implementing the system as a web application will enable us to come close to a generic type of user. Web application can anyway run both on Internet and Intranet, giving us the possibility of using both Internet and LAN networks for our solution.

4.4 System Architecture

There are three main distribution patterns for web application: Thin Client, Thick Client and Web Delivery.

The tables below show advantages and disadvantages of these patterns in connection to the system.

	Thin Client Web Application Pattern
	Connection is cheaper because is used only to receive the results.
	Inflexible interface. In Thin Client the interfaces are fixed, there are no functionalities.
	The response time depends exclusively on the server and the size of the database.

	Thick Client Web Application Pattern
👍	Adds functionality to the Thin Client Pattern
👍	Supports sophisticated interfaces
👎	The response time depends on the size of the database and the capability of the server

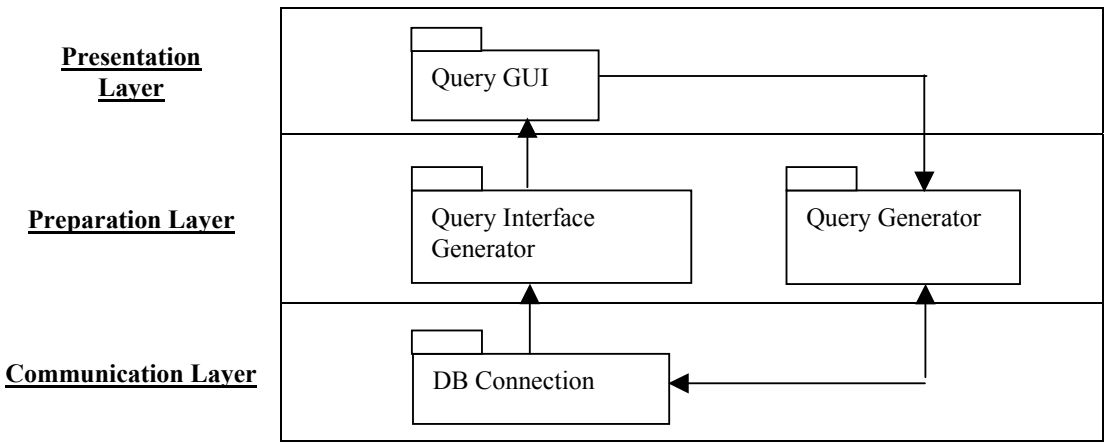
	Web Delivery Web Application Pattern
👍	Components are automatically loaded in the browser when needed
👎	Need longer time connections.
👎	Inflexible server structure.

All transaction passes by the Server and through the DBMS, therefore the response time of each of those design patterns depends mostly on the size and capacity of the Server and of the DBMS, even though the functionality are placed in different places.

However, as this system will have to communicate with existing servers, we assume that the User Company has a server that can handle the amount of users that the company will allow access. The Thin Client is a pattern that does not fit with our system because it goes against the requirement of a flexible interface, see *Requirement Set: Software Requirement Specification, section 4.8.1*. The Web Delivery pattern will lead us to eliminate the functionality from our client, and we cannot pretend the User Company to have a server with already all the functionality installed. The ideal solution will be to use the Thick Client Pattern, it will allow us to interface the database server and transfer the needed functionality in the client.

4.5 System Layers

As flexibility and reusability are high requirement of the system, we decided to divide the system in layers. Making independent layers will give us the possibility of creating a more flexible architecture; every time a change is needed, it will be possible to change only the interested layer, while the others will work anyway. The different layers could then also be divided between the server and the client depending from the architecture chosen.



4.5.1 Presentation Layer

The Presentation Layer receives raw data from the Preparation Layer and formats it in a presentable way to the user. In formation about the querying criteria that can be used are received from the Preparation Layer, and the Query GUI will change according to them.

4.5.2 Preparation Layer

The Preparation Layer takes care of preparing the information transaction from and to the Presentation Layer. The Query generator reads the querying criteria sent by the Presentation Layer and submits them to the Communication Layer, in an understandable format. The Query Interface Generator receives the database structure from the Communication Layer and send to the Presentation Layer information on how the querying can be presented to the user.

4.5.3 Communication Layer

The communication layer takes care of open and closes the communication to the server and to send and receive the data from the database.

4.6 Solution Options for the Query System

Three different options, concerning the architecture of the system, has been offered to the User Company:

- Complete XML Solution
- SQL Solution
- Protocol Solution

4.6.1 Complete XML Solution

This solution is based on XML to query the server and transport data.

Server

- Sends DTD file describing the database structure
- Reads the query in XPATH querying language
- Sends the result of the query in XML

Client

- Creates GUI based on the structure described by the DTD file
- Creates and sends XPATH query to the server
- Visualise the XML result using XML schemas

Advantages

- The system will not care what kind of DBMS is used
- There are existing tools for creating DTD files from databases e.g. *"DB/XML Vision by DataMirror Corporation"*
- Easy to configure to a DBMS

Disadvantage

- The DBMS must support XML
- The server must understand XPATH

- Complex solution, it requires more implementation time

Customer Possible Judge

The Customer might like the short configuration time, but as our costumer is very reluctant to reveal its database structure then this solution might discourage them.

4.6.2 SQL Solution

This solution is based on SQL to communicate with the server.

Server

- Sends a query in SQL representing the database view for the interface.
- Sends a list of fields where the search can run.
- Sends the result of the query.

Client

- Creates a GUI from the list of fields coming from the server.
- Creates a Query on the Query using the searching criteria.
- Receives the result and displays it

Advantages

- Easy to implement
- Well-known technology and tools

Disadvantages

- Bound to RDBMS and therefore limiting the flexibility of the system and would not allow the system to be connected to different database systems like e.g. Object Oriented DBMS

Customer Possible Judge

Known technologies are always preferred, but the Customer might not like to send information about the database structure. The Customer might not have a RDBMS.

4.6.3 Protocol Solution

This solution is based upon the creation of a standard protocol for communication with the server.

Server

- Sends a list of fields where it is possible to search.
- Creates a query from received search criteria.
- Sends the result in XML

Client

- Creates a GUI from received searching fields.
- Sends search criteria.
- Visualise XML result using XML schemas.

Advantages

- Flexible database interface

- No need for any database structure.

Disadvantages

- DBMS must support XML or translating routine must be implemented.
- Server has to have the querying routine.

Customer Possible Judge

The customer would probably like the idea of not giving the database structure and not been bound to a DBMS because then they can change the database structure and system as they want. However, it might not like to have to implement querying routine in the server.

4.6.4 Tools, Technology and Knowledge Level

The following table is a summary of the technologies needed for the different proposed solutions, and in relation to it, we also show the level of knowledge of the team in relation to those technologies. The reason for compiling such a table is to help us in the decision-making. For example, if the speed of development is an important issue in the project we could deduce from this table that the SQL solution will improve the development speed because the technologies involved are well known.

Protocol Solution	Level Of Knowledge
ASP or Java™ XML XSL or CSS schemas	Medium
Complete XML Solution	
XML DTD XSL or CSS schemas XPATH Java™ or ASP	Low
SQL Solution	
SQL ASP or Java™	High

4.6.5 Times

The following table is also used as help in the decision-making, it shows the time each solution would need in order to add a new database to the system and the implementation time behind each solution. By “Us” and “User”, it is showed the time the user needs to spend in adding the database, and the time the development team needs to implement eventual changes.

The times are measured in day units; eventual extra times are expressed in words.

Time to add a new database						
Phase	**SQL Solution**		**XML Solution**		**Protocol Solution**	
	Us	User	Us	User	Us	User
GUI	0	0	0	0,5	0	0,5
Querying	0	0	0	0	0	0
Interface	0	0,5	0	0,5 + tool	2 or more	0
	Implementation time					
	3-4 Weeks		4-5 Weeks		3-4 Weeks	

4.7 The Architecture Selection

4.7.1 User Company's Answer

The Project Proposal Document that was sent to the User Company had 3 possible architectural solutions to the system; the purpose of it was to allow the User Company to select the most suitable solution for them.

The User Company did not choose any of the solutions sent but it used some of the ideas and gave us this in return:

1. *The system consists of a client application, an HTTP server and a DB server.*
2. *XML should be used for communication between the HTTP server and the client.*
3. *SQL should be used for communication between the DB server and the HTTP server. (DB server may, or may not, be able to deliver query result in XML.)*
4. *Intranet will probably be used (but the system should also work using the Internet).*
5. *The client program should be designed so that it could be used by a keyboard with minimum number of keys (perhaps only number and functional keys; letter keys could be displayed on the screen).*
6. *Tests will be done using a regular laptop. If the project goes well, EB may decide to buy a special car computer.*

In the following section, we describe the architectural design that finally was decided to be implement, emphasising on the changes made to the User Company's proposed solution.

4.7.2 The Chosen Architecture

The system will run on a client-server architecture, where the communication between the system and the database will be run at the server side. Communication between the server and the clients will be done using Java™ streams[1]. A database driver such as ODBC[2] or JDBC[3] will handle communication between the server and the database.

In the following section, we try to describe the reason for changing some of the architecture choice of the User Company into what is to be implemented.

Java™ Stream vs. XML

During the first meeting, we had the impression that we were suppose to implement a client application that would be able to connect to a server owned by the User Company, but during the requirement research, we discovered that the server implementation was actually a part of our system. For that reason we decided to use a more suitable communication means between the clients and the server, which will make it easier for us to implement a flexible structure. We then thought that XML would be good communications means that most application uses, but as it has already been decided to implemented both the server and the client with Java™, we found it more convenient to use the already build in communication structure of Java™ (Java.io). By doing so we will use an already tested and a reliable communication structure that will help us decrease the implementation time, and therefore provide the Demo software in a shorter time.

[1] Sun Microsystems Java™ SDK 1.4.0, package Java.io

[2] Microsoft Corp.

[3] Sun Microsystems

Server vs. HTTP Server

Choosing not to implement the server as an HTTP server, but rather as a normal server, is based on a functional reason. An HTTP server gives responses only to requests, and this does not fit the requirement of the system. The System is supposed to be able to send messages between a Central Operator and a Mobile User, but an HTTP server will not be able to dispatch a message to a client that hasn't send a request to retrieve the message. Again, the same problem will rise when a Central Operator will try to send a query result to a Mobile User. The main problem is that an HTTP Server does not keep the connection with a client, but by using a normal server, it will be possible to keep the connection to the client and exchange information with it at any time.

5. Software Architecture Document

5.1 Introduction

5.1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions that have been made on the system.

5.1.2 Scope

The scope of this document is to give a general overview of the architectural structure of the Car Tracking System.

5.1.3 Overview

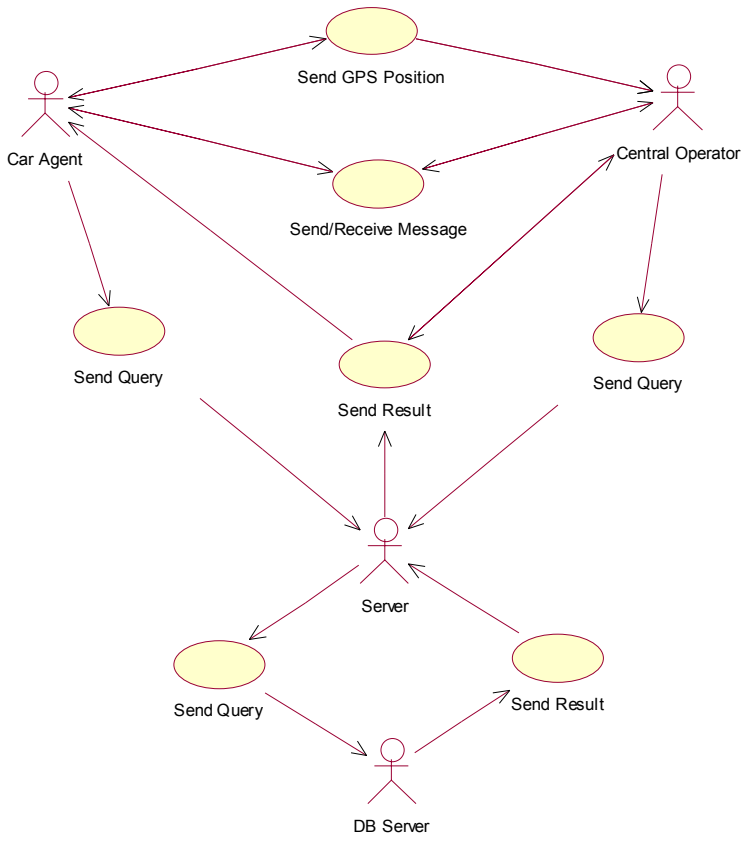
This document represents the general architecture of the Car Tracking System in the form of Use-Case view, Logical view and Deployment view.

5.2 Architectural Goals and Constraints

The most important objective of this system architecture is the flexibility and independency from the database Server. The designed architecture should give the maximum flexibility and the possibility to transport the system to other Users with the minimal amount of changes.

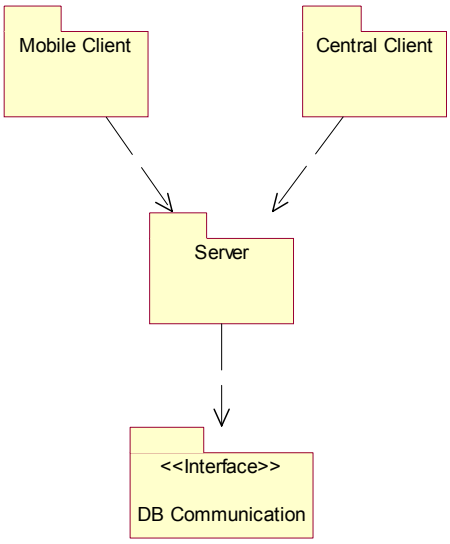
5.3 Use-Case View

RUP® suggests to include all significant Use-Cases in this document, but as we treat the system as a group of three subsystems we decided to show in this document only the views that will allow us to have a simple and general overview of the system, and keep the ones specific of each subsystem in separate documents for each subsystems.



5.4 Logical View

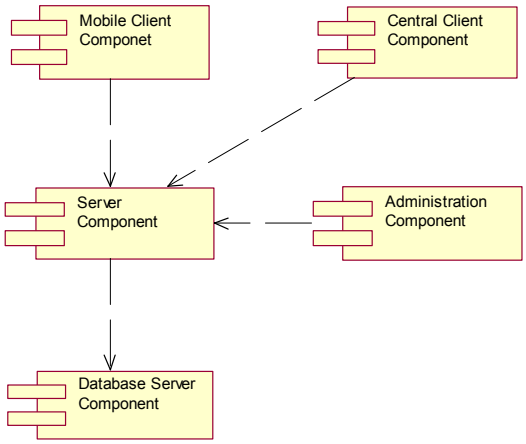
The Car Tracking System is composed of two clients, a Server and a Communication Interface with the Database Server. The Central Client is the client that has all the functionality connected to security and connectivity to the DB. No special requirements have been given from the User Company about the Central Client; the User Company actually does not ask for more than a GUI for the server application, but to ensure a smoother move to further implementations we decided to implement it as a client. This will eventually enable the system to have more than one Central Operator working on the system. The Mobile Client will be used remotely and will connect to the Server for messaging and querying services. The Server will be some kind of router that will manage the exchange of messages and data. In addition, the Server will also be used to exchange information for the GIS between the tracking subsystems.



5.5 Component View

Looking closer into the subsystems we found out that they all share the same component structure, except only the Query System has the connection to the Database Server Component. Therefore, we have included the component view here instead of going through it with every subsystem.

The following diagram shows the division of the classes into components and the relation between them.



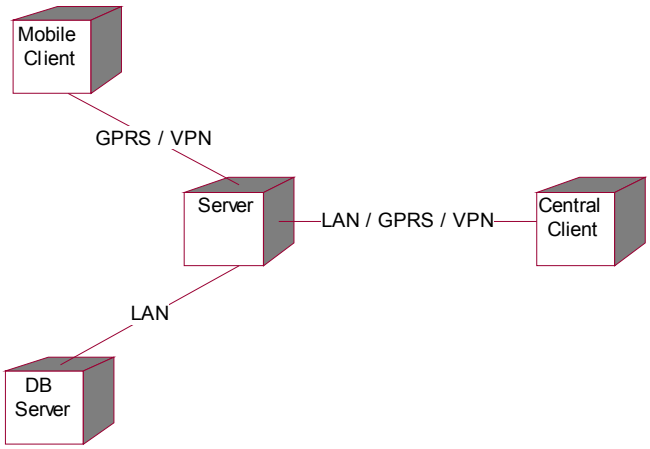
During the different iterations we will run the design section of each component, setting up communication interfaces between the components in order to create independent components and increase the flexibility of the system.

Notice though that the Database Server component is just a representation of the Database Server owned by the User Company, and therefore there will be no design of this component. Neither will we do any design of the Administration Component due to the scope of the demo version.

5.6 Deployment View

The Central Client, the Server and the DB Server will be installed in the same LAN, while the Mobile Client will access the server remotely either via the Internet or directly (VPN) using the GPRS wireless technology.

As shown in the diagram it is also possible to connect the Central Client through the internet using a VPN or a GPRS device, this allow us to implement the Mobile Client and Central Client as a single application and control the access level with an authorisation mechanism.



6. Conclusion

This phase of the project has been one of the most long and with the most number of decisions. Starting with making researches of how the architecture of the system could be, we compiled an architecture proposal document for the User Company. Differently than expected the User Company did not choose one of the solutions send by us, it compiled a new solution, which was a mixture of the solutions we proposed. With a few changes, an architecture design has been finalized in the expected time. During the successive Iterations, there has never been the need to change anything in the original architecture design, beside the addiction of functionalities to the system.

All the documents and artefacts selected for this set has been compiled using the templates coming from RUP®. Though we think that the templates from RUP® are of very good quality, we found that in many of them there were information that had no significance to the project, therefore we decided to tailor the documents, and to add few of the extra information that we found more useful for the project, or as in some cases create our own documents. An example of this is the *General System Architecture*[1]. As previously explained, we decided to give the User Company the possibility of choosing the architecture for the system out of a short selection. Information relative to the architecture choice has also been added, including some technical details such as the implementation time needed if the system is to configure to a new database.

[1] Page 73

Implementation Set

1. Introduction

It is always nice to see something being created and coming to life. That is what the Implementation Set is about, start seeing something being born, something else than documents.

This chapter does not explain implementation decisions, as these decisions are handled in the chapters of each subsystem, however we discuss here the implementation restriction associated to the demo version of the system, and some refactoring issues that should be done.

Here is where XP comes into the picture and we try to explain how those activities came into practice in our project and explain how we used them.

2. Implementation Restrictions

2.1 Introduction

2.1.1 Purpose

To address issues which were considered to be out of the scope of the demo version due to various reasons. This will give future developers of the system an idea of things that have to be added to the system and even sometimes an idea of a way to add them.

2.1.2 Scope

The scope is the limits of demo version of the Car Tracking System.

2.2 Limitations

2.2.1 System Status Control

By system status control, we mean all the functionalities related to the reliability of the system. Those functionalities include the handling of Server, Database Server and Client crashes. The system should be able to handle those crashes by checking the status of the connection. The only one of those functionality implemented by the server is the monitoring of communication crash between the server and the client, the crash is though only implemented at the server side.

2.2.2 System Administration

By System Administration, we refer to all the functionalities related to the administration of the server and of the users of the system. Examples of those functionalities are the adding and removing of database connection to the system. The implementation of those functionalities should though be postponed until the analysis of the Administration System Component has been done.

2.2.3 GUI Looks

The GUI looks has been selected as a low priority feature because it does not address any special and central feature of the system. Though requirements and importance of the GUI have increased, we decided to keep the GUI analysis to a later stage then the Demo in order to keep the focus on the functionalities of the system. However, all known requirements regarding the GUI have been documented.

2.2.4 Authentication

Authentication functionalities will allow the system to identify users and load different functionality to the system depending on the access level of the user. For the demo version, the two types of client have been implemented as two different applications, but they should have been implemented as a single application that changes functionality depending on the access level of the user. That would have made it possible to install only one application at every workstation.

2.2.5 Information Security

Information security is an important issue of this system, but it is out of the scope of the demo version, because we do not believe it would help in displaying the essential of the system, as the security features are just an add-on to the system.

Furthermore, we decided not to implement it because as everything is implemented using objects, it will be relatively easy for future programmers, to change the class that needs security without needing to change other parts of the system. Security features can be implemented using the Java.security[1] and the Java.crypto[1] packages provided by Java™.

2.2.6 Acknowledgements

Showing acknowledgements for transactions is needed in order for the users to know that sent information are reaching their destinations as they are supposed to. For example when the Central Operator sends a Query Result to a Mobile User then there is no way for the Central Operator to know if the Query Result reached its destination or not. We suggest though, instead of displaying an acknowledgement every time a transaction has taken place then an error message is to display instead if a transaction fails with in a set timeframe.

3. XP Practices[2]

3.1 Pair Programming

Pair programming is one of the technique that we imported into RUP® from XP. Two programmers sitting at the same computer sharing their ideas and concentrating in quality, create the code. Following the guidelines of XP, this will help the programmer in the long run, because as two persons are overlooking the code it will be easier to spot errors and incompatibility with the architecture.

During this project, every time code was involved we gathered around one computer and started programming. This turned out to be a very good process, while the programmer that was typing was concentrated in how the function was working, the programming parted would concentrate more on how the function could fail and on how this function would fit in the system.

3.2 Refactoring

Refactoring is another of the practice that we imported from XP. It consists in the continuous search for a better way of designing and programming. It does not mean that one should spend time on redesigning the whole system, but every time one finds a better way of programming a function or a better design, he/she should be free of changing the code or de design to the better one.

During this project refactoring was a very important part. The whole system has been designed with the idea that it will be changed, therefore we kept a simple design and code, which would be easy to change. As the implementation of the system, as well as the whole project process, has been divided in iterations, we came across the same classes different times, and every time we refactored the code and the design of the system to better fit the new requirements that were discovered.

3.3 Collective Ownership

As for pair programming and refactoring, collective ownership is also a practice taken from XP. It consists of sharing the code between the programmers, meaning that anyone can change or add functionality to the code, without asking permission to anyone. In order to keep the changed code consistent with the rest of the system, automated unit tests are created, and only code that passes the test 100% can be released.

[1] Sun Java™ SDK 1.4.0

[2] Methodology Resources

In this project, we extended the concept of collective ownership not only to the code but also to all the documents that were part of the project. The code and the documents were hold in a share driver and could have been accessed and changed by any of the members of the team. Differently then for the code, were unit test could have been made to check the consistency with the system, the documents needed something to show had been made. Therefore we decided to use a review history, which will show what changes had been made to the documents.

3.4 Continuous Integration

This practice, also coming from XP, tells us to continuously integrate our code. Every pair of programmers should work in the latest version of the system, and integrate often. Also this practice from XP has been successful in our project, instead of implementing the three subsystems as separate systems and then integrate them at the end, we integrated the code to the latest version of the system, giving to the Company the feeling that the system was growing.

This practice also helped us in the continuous research for new and clearer requirement. Every time we had a demonstration of the system with the User Company, it was possible to show the system as a whole. In this way, the User Company was facing the whole system every time, letting it have the chance to think about requirements that might have been overviewed in previous demonstrations.

4. Conclusion

The implementation section of the project has been of very big interest, because we were to test the new working practice suggested by XP. At the start, it was a bit odd to be two programmers in front of a single computer, but the uncomfortable feeling did not last too long. We soon learned what actually was the job of each programmer, and then the process went smoothly and less and less bugs were found in the system as the process went.

While programming, each programmer would look at the code in a different way and from a different angle. This helped us to create a more simple code and design and to avoid the usual logical mistakes that often come from the rush of programming fast.

The iterative process and the refactoring practice helped us looking more than one time at the implemented classes and creating a more flexible system.

The only real document that RUP® proposes for the implementation set, is the Implementation Build Plan[1], but as this plan was not going to be used, it has been cut out during the tailoring of the methodology. We found it; instead, more useful to create a document, which would contain all the important implementation decision taken during this section. This document will help anyone who is new to the project, to understand the important implementation issues met during the building of the system, without the need of looking through the code itself.

[1] RUP® Framework, support documents and tools

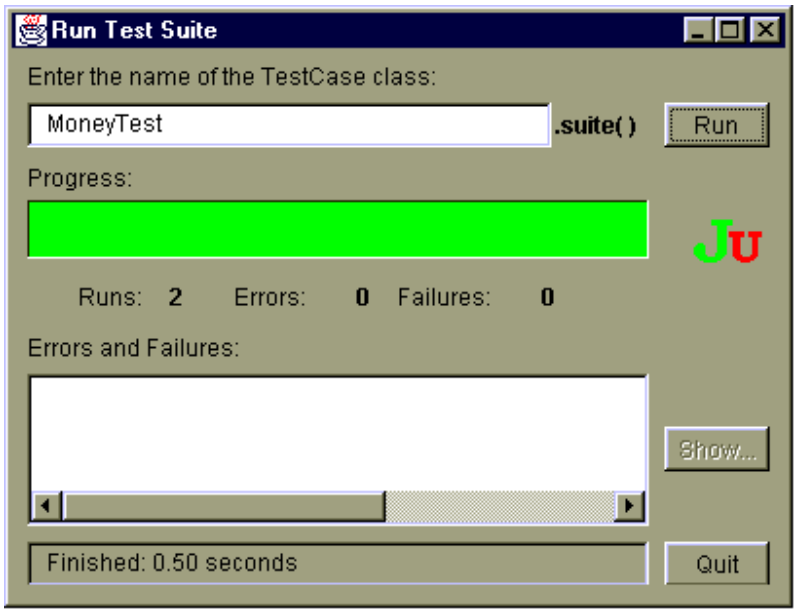
Test Set

1. Introduction

In order to produce a quality product we need to test it, one way or another. The Test Set is very small and here we only cover JUnit™ and how we used it during the development process. However, we produced a Test Plan for each subsystem in order to identify the software components that should be tested, list the test strategies need for it and review the results of it. These documents are located in the chapters related to each subsystem.

2. JUnit™

JUnit™ is the automated testing tool we used during the project. The tool is a framework for automated unit tests, created by Kent Beck and Erich Gamma. For each class a test case is created and it will test the class methods. Giving an expected value or event that the method should return or react to, and compare it to the actual behaviour of the method makes the tests.



This tool allows the programmer to create fast test cases that does not involve any changes in the actual system code, by creating the test cases as stand alone classes.

For many of the most logical classes, which had computing functionality, we had no problems in creating and running the test cases. The lack of experience with this tool has though given us some problems in designing particular tests, in which case we decided to use manual tests. We though feel that this tool would have been of much greater help to our project if we had the time to get experience in test case designing. The web documentation and the number of add-ons for this tool, such as GUI testing, make us confident in the usefulness of this tool in our future projects.

3. Conclusion

Testing was an important part of our project, deciding to follow the XP practice; we had to test continuously during the implementation and the integration of the system. As suggested by XP we created automated unit test, but we found it very hard to test the actual network communication between client and server. Therefore, we were facing the problem of deciding whether to use time in finding a way to test the communication using the automated tool, or just test the communication in a different way. Due to the timeframe we were working on, we decided not to create the automated test, but to set up some use-cases based test for all the part of the system that could not be tested automatically.

The three test document that we have choice to use from RUP®, turned out not to have a lot of information as separate documents, therefore we decided to combine them into one single document, that in our project case, gave an overview of which tests were made, how they were made and the result of them.

Query System

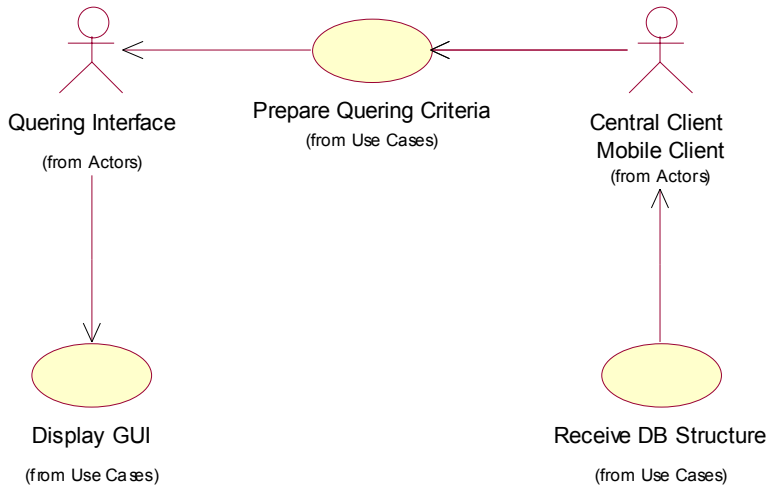
1. Introduction

This subsystem of the Car Tracking System is used to allow the users to connect to the central database and be able to run queries on it. The main functionality of this system is actually not the ability of sending queries to the database, but the ability of connecting the system different databases and having the system's GUI adjusting to the new database automatically. The system furthermore allows the Central Operator to send a specific Result directly to one or all of the Mobile Users. All the transactions pass through the Server.

The chapter starts by describing the most essential use case for the subsystem, other use cases can be found in Appendix C. Thereafter we go in to details of the architecture of the subsystem in the Software Architecture Document, following by the Implementation and the Test documents. Software requirements specifications for the subsystem have been captured and described in the Software Requirements Specification in the Requirements Set[1].

2. Requirements Set: Use Case Specifications

2.1 Generate GUI



2.1.1 Details

Description
The Query Interface generates a GUI from the information given by the Central/Mobile Client. The information is created from the database structure and passed to the Central/Mobile Client.
Basic Flow
• The user selects a database to search in • The Central/Mobile Client receives the DB Structure • The Central/Mobile Client fills-in the info for searching GUI criteria • Query Interface reads info • Query Interface creates the GUI • The Use-Case ends

[1] Page 64

Alternative Flows
1. Structure Invalid *The DB structure passed to the Central/Mobile Client in not a valid structure. A message displaying the fatal error is displayed. The Query System closes.*
Pre – Conditions
None
Post – Conditions
A Graphical User Interface is created suiting the DB structure
Special Requirements
None
Frequency
Every time the query window is opened
Primary Actor
Querying Interface
Secondary Actor
Central/Mobile Client
Secondary Use-Cases
None

3. Requirement Set: Software Requirements Specification

See *Requirement Set; Software Requirements Specification*.

4. Analysis & Design Set: Software Architecture Document

4.1 Introduction

4.1.1 Scope

The scope of the Software Architecture Document is to give different views of the Query System, (a subsystem of the Car Tracking System), with enough details to give the programmer all the information needed to start the system implementation.

4.2 Architectural Representation

The architecture of the Query System is represented in the form of a Use-Case view, Logical view, Deployment view and Implementation view.

4.3 Use-Case View

4.3.1 Use-Case Realisations

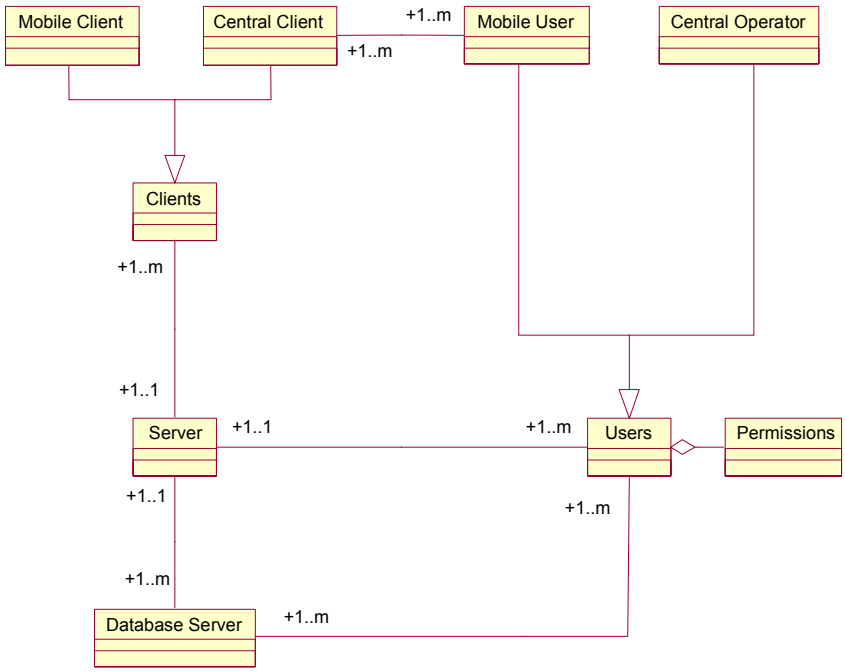
The Use-Cases realisation can be found in *Appendix C*

.

4.4 Logical Views

4.4.1 Analysis Model

The following diagram shows the Logical view of the system; it does not have any details because this is left to the design of each component. Following the diagram, we will give a brief description of the overall system, explaining in a general manner the classes and the connection between them.

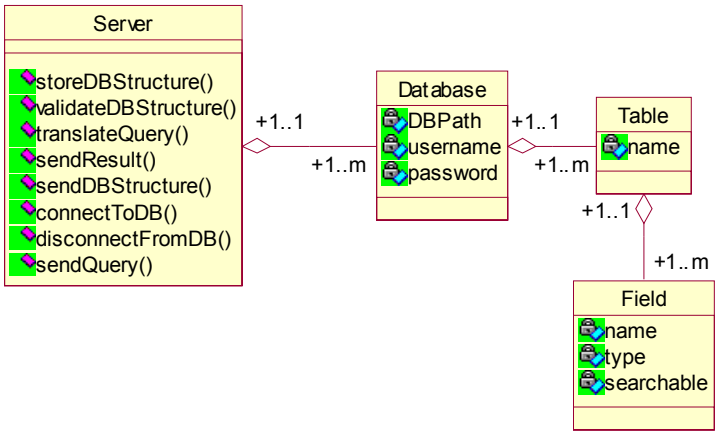


The most important classes shown in the diagram are the Client, the Server and the Database Server Classes. The Client classes describe the remote applications that will connect to the Server for data services. The Server handles the requests and sends/receives data for the Database Server class. The Database Server class itself is a representation of the Database Server owned by the User Company. Information about the users are described the Users classes. Of particular interest is the association between the Users classes and the Database Server class; every user has particular "*Permission*", on the data that can be accessed, therefore the association represents the connection between a particular User and the Databases that he/she is allowed to access.

4.4.2 Design Models

Server Component

The Server component is one of the central and more critical parts of the system. It has to handle all the requests from the Clients and forward them to the User Company Database Server.



The Component is made of two classes; one is the Server itself and the other, “Databases”, is the representation of the databases connected to the Server. For security reason the User Company does not want the Server to know directly the structure of the databases, therefore the database structure is passed to the Server as an external file. The structure described in the file will be held in the Databases classes, which will contain all the data relative to those databases. The database structure held in the “Database, Table and Field” classes is then used by the system as a reference to how the database is structured.

Functions

N.	Function	Type	Specification	Complexity	Uncertainty
1	Server.sendResult()	S	N	1	1
2	Server.sendDBStructure()	S	N	1	1
3	Server.storeDBStructure()	U	N	1	1
4	Server.validateDBStructure()	C	S	3	3
5	Server.translateQuery()	C	S	3	2
6	Server.connectToDB()	C	I	1	1
7	Server.DisconnectFromDB()	C	I	1	1
8	Server.sendQuery()	S	N	1	1

Legend

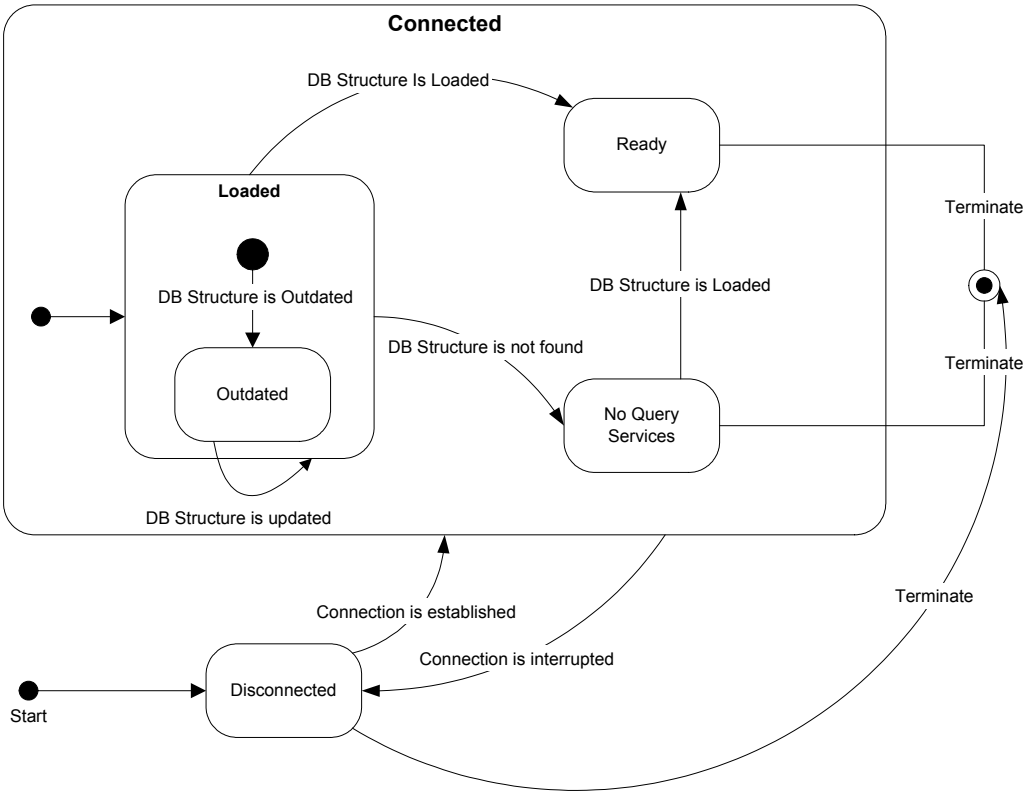
Type	S = Signal U = Update C = Computing
Specification	N = Name I = Implicit S = Specification
Complexity	1 – 5
Uncertainty	1 – 5

Function Specification

See *Appendix D.*

Mobile Client Component

The Mobile Client Component is one of the two Client applications of this system. This component is used to send queries to the server and display the result. In order to have a better understanding of what are the events around this component, we used a State Chart Diagram of the MobileClient class.

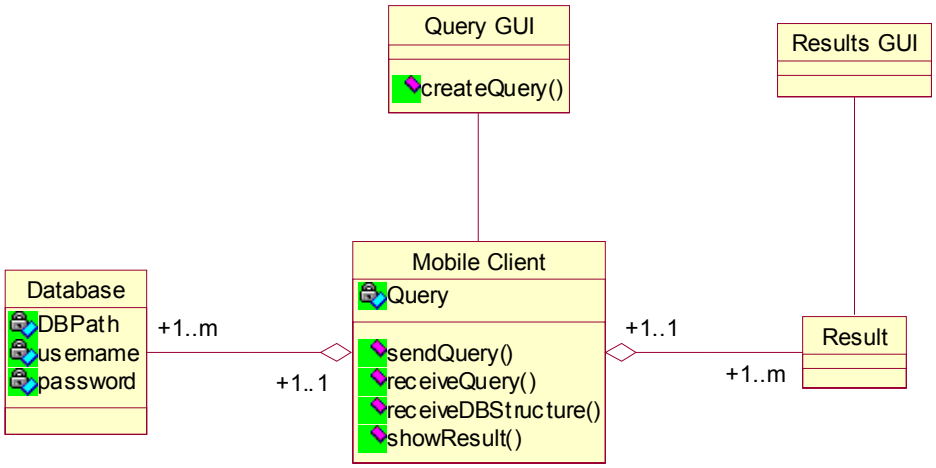


The above diagram assisted us in learning about the different events that are around the Mobile Client, and gave us hints on possible error situation such as the crush of the server. The Mobile client, once started, will try to connect to the server and, if the connection is successful, it will check if the Database structure is up to date. If not then the client gets an updated structure, before entering the ready state; otherwise, it goes straight into that state. The No Query Services is an error state, which describes the absence of a database structure; therefore, the query services cannot be performed.

The State Chart Diagram also brought up a discussion about how and when should a Client check for updates in the Database structure. We came out with two solutions that can be considered. However, it should be kept in mind that we did not do any thorough investigation into these solutions; neither do we plan to implement this feature due to the limited scope of the demo version.

The first solution would be to introduce a “date & time” attribute in the Database classes, so that every time that a Client connects to the Server can check if the Database structure in the Server has the same date. However, this solution implies communication with the server at least once a day from each client.

Considering that the Database Structure will probably not change so often, the second solution suggests having a list of the client that do not have an updated database structure. Once a client connects, the server will check if the client is outdated, and send the updated version if needed.



The above diagram is the design model of the Mobile Client. The central class is the Mobile Client class, which handles the communication with the server. The Database class is the representation of the database. The data for the Database class comes from the Server and becomes updated every time the structure changes. The result class holds the results coming from queries sent to the server. We set it as a class because the results do not have to be only displayed but also stored; a log of all the results must be kept for the login time. Because of the possible large size of the results, we decided to keep the log to a maximum of 50 results. The "ResultGUI" is used to display the Results received from a query and to show the Result log. The "QueryGUI" is automatically created from the information stored in the Database class, and allows the users to formulate queries.

Functions

N.	Function	Type	Specification	Complexity	Uncertainty
1	MobileClient.sendQuery()	S	N	1	1
2	MobileClient.receiveResult()	S	N	1	1
3	MobileClient.receiveDBStructure()	S	N	1	1
4	MobileClient.showResult()	C	N	1	1
5	QueryGUI.createQuery()	C	S	2	1

Legend

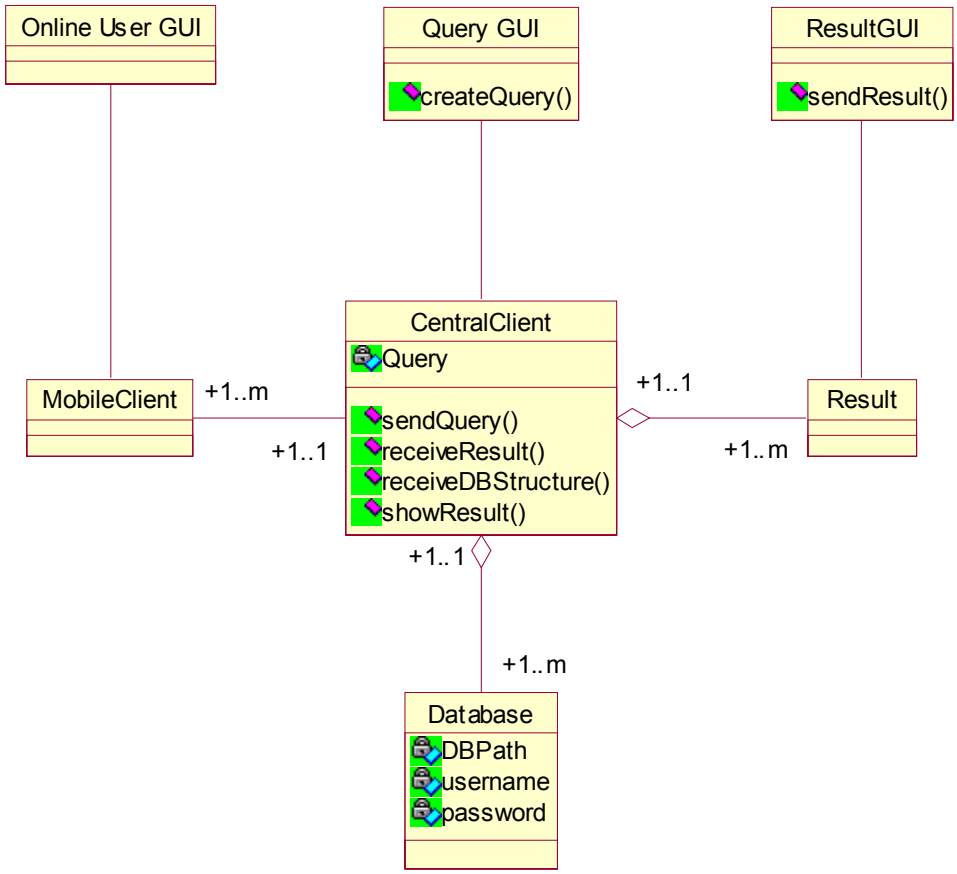
Type	S = Signal U = Update C = Computing
Specification	N = Name I = Implicit S = Specification
Complexity	1 – 5
Uncertainty	1 – 5

Function Specification

See *Appendix D*.

Central Client Component

All that has been stated for the Mobile Client Component is also valid for the Central Client Component, and therefore we will limit ourselves to display the design model and explain only the differences. Only extra functionalities will be displayed.



The Central Client has two more GUIs, than the Mobile Client. The “OnlineUserGUI” enables the Central Client to see all online Mobile Clients. The “ResultGUI” is the same as the one from the Mobile Client with the addition of the option of sending the found Result to a specific Mobile Client.

Functions

N.	Function	Type	Specification	Complexity	Uncertainty
1	ResultGUI.sendResult()	S	S	1	1

Legend

Type	S = Signal U = Update C = Computing
Specification	N = Name I = Implicit S = Specification
Complexity	1 – 5
Uncertainty	1 – 5

Function Specification

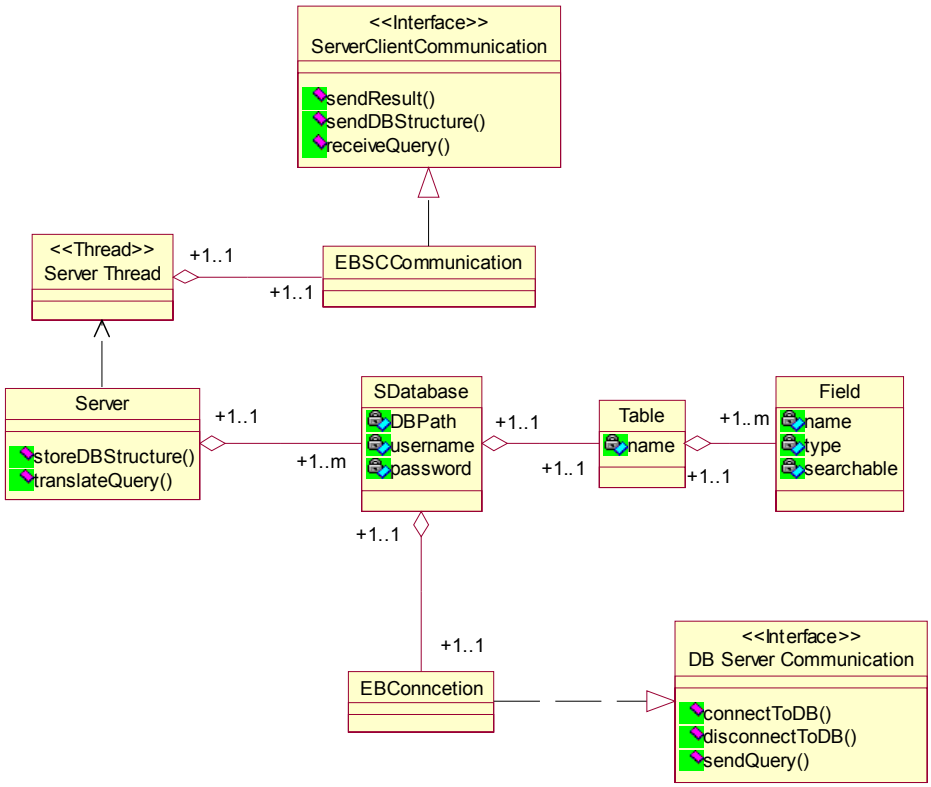
See *Appendix D.*

4.5 Deployment View

See *Analysis & Design Set; Software Architecture Document.*

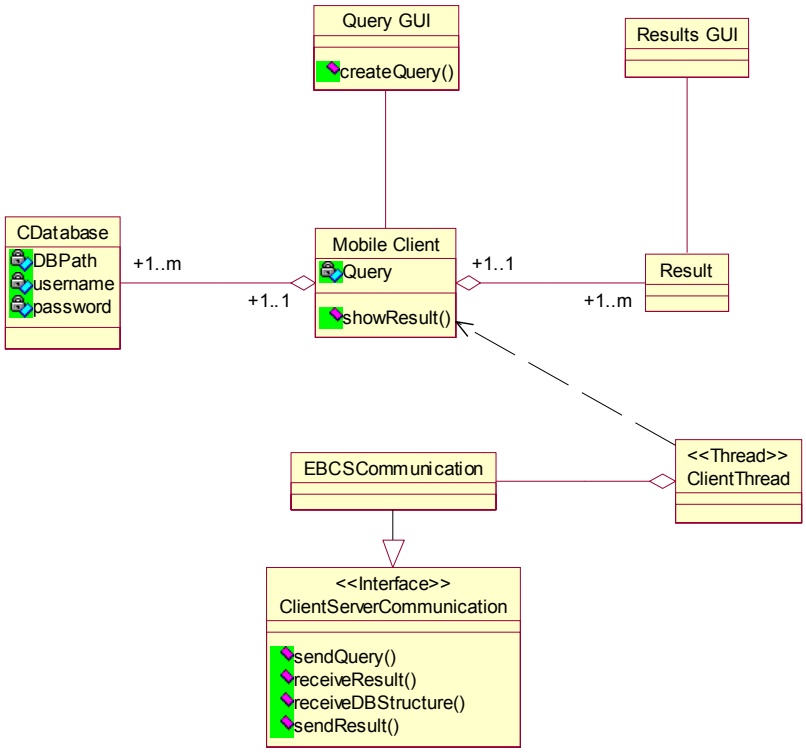
4.6 Implementation Views

4.6.1 Server Component



Client Communication and DB Server Communication are the interfaces that connect the Server Component to the Database Server Component and the Clients Components. To make the Server able to handle more than one request at the time we decided to implement the Client Communication interface with a Thread, which will then serve the clients. The Database class will be loaded in a java.vector[1] class and will be persistently stored in a file. By doing so we will need the information of the database structure only once, and by loading it into a Vector it will improve the performance of the algorithms that are using this structure (e.g. validateDBStructure()). Note that vector was chosen because we were very familiar with it but other, like Arraylist could also have been used.

4.6.2 *Mobile Client Component*

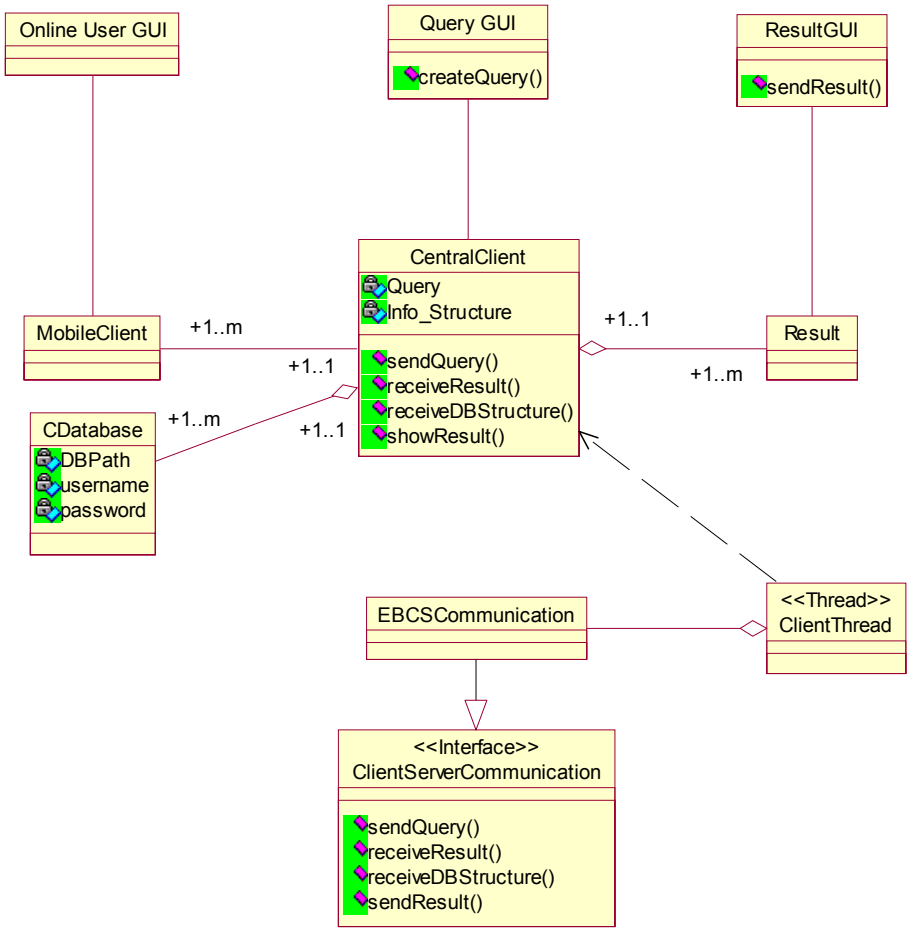


Notice that in this diagram and in the Server Component Implementation diagram, that the Database class, in the design section, has changed name. The reason for this is that the information stored in the Database class in the server is more confidential (e.g. DB Server password), and should therefore not been sent through the network for unnecessary reasons. Our solution to this is has been to make the Server Database Class an extension of the normal Database class. By using a generalisation, we also can construct the Database class for the client with a bit more flexibility, allowing future implementation to change the class without interfering with the server classes.

The ClientServerCommunication interface describes the methods that will be used for the communication. The EBCSCommunication class as part of the Mobile Client class then implements the interface. To ensure that the user can interact with the different GUIs without being interRUP®ted from Client-Server communication, all incoming communications are handled by the ClientThread class, which runs as a separate thread.

[1] Java™ SDK 1.4.0 java.util package

4.6.3 Central Client Component



The implementation comments for the Mobile Client and the Central Client are the same; the only difference to mention is that the Central Client implements the “sendResult()” method of the ClientServerCommunication interface.

5. Implementation Set: Implementation Document

5.1 Introduction

5.1.1 Scope

It is important to realise the scope of the demo version of the Query System before going any further. The demo is implemented so that it can show the main and most important feature of the system. Those features are the sending of query, the receiving of results and the sending of results from a Central Client to a specific Mobile Client.

5.2 Implementation Issues

5.2.1 Observer/Observable Pattern[1]

The "Central Client" has the functionality of showing all online "Mobile Clients". In order to implement this functionality we decided to use the Observer/Observable pattern. We chose the "CentralClient" class as an Observable object because it is the class containing the information about the Mobile users online. As Observer, we chose the "OnlineUsersGUI" class, which displays the online users on a JList[2] component.

```
public class OnLineUsersGUI extends JPanel implements Observer{
        .
        .
        public void update(Observable o, Object arg){
                updateListModel();
                this.repaint();
        }
        .
        .
}//end class
```

However, doing so, we met the problem of trying to use multiple inheritances in Java™; the "CentraClient" was already a generalisation of the class "Client" so it could not be a generalisation of Observable[1] (as it must be in the Observer/Observable pattern). To solve the problem we decided to create a class that would hold the information about the users ("Users" class), this class could then inherit from Observable without any problems.

```
class Users extends Observable{
        .
        .
        public void add(User user){
                usersVector.add(user);
                setChanged();
                notifyObservers(usersVector);
        }
        public void remove(User user){
                this.usersVector.remove(user);
                setChanged();
                notifyObservers(usersVector);
        }
        .
        .
}//end class
```

[1] SUN Java™, JDK 1.4 Java.util package

[2] SUN Java™, JDK 1.4 Javax.swing package

5.2.2 Mobile and Central Client Information

Every time the server receives a connection from one of the clients, the server checks if the client is a Mobile Client or a Central Client, and place it in the respective Vector[1]. The Vector actually stores a reference to the Server Thread that communicates with the specific client. The reason for holding this connection is to be able to refer to one specific client. The following code is an example of how those Vectors has been used.

```
public void sendUsers(int CCID){
  if (mobileThreads.size()!=0){
    for (int i=0;i<mobileThreads.size();i++){

      User tmpUser=((ServerThread)mobileThreads.get(i)).getUser();
      ((ServerThread)centralThreads.get(CCID)).sendUserLOGIN(tmpUser);

    }//end for
  }//end if
}//end function
```

The above function is used to send information of all Mobile Clients online to a specific Central Client. "`mobileThreads`" and "`centralThreads`" are the Vectors containing the references to the Server Threads working on Mobile and Central Clients. In this case, the "`mobileThreads`" Vector is used to read the User information of all Mobile Clients and then send them to the specific Central Client in the "`centralThreads`" Vector at position "`CCID`"

5.2.3 Communication Implementation

Following the implementation view, created in the design phase, the communication interfaces ("ServerClientCommunication", "ClientServerCommunication") were supposed to be implemented by the running Thread of the client and the server, but this would have implied changing the Thread itself in case of future development. Therefore, we decided to implement the two interfaces in two different classes and have the Threads using them. By doing so we improved the flexibility of the system, if changes will be needed to the way the system communicates the developers will only need to change the interfaces implementing classes and the whole system will still work.

5.2.4 Communication Performance Problems

At the first release of the software, all tests has been run in a LAN environment, where the communication speed did not allow us to notice any possible problems with the communication performance. After the first release we tested the software in a real-life environment, where the client connects to the server using a GPRS enabled mobile phone, the communication speed was drastically reduced but still acceptable for small size packets communication, such as text only fields.

The problem rose once we included an Image object in the communication object. After testing the system by sending different kind of data, we got to the conclusion that the major problem was due to the unstable connection held by the GPRS phone.

[1] SUN Java™, JDK 1.4 Java.util package

We think that if the connection results in errors while sending an object, the object will then be resent, and this could be the reason of so many delays because the Objects that are sent by the system are too big, so they have a bigger probability of resulting in errors. Therefore, we decided to split the Object sent (instance of Result class, holding all results of a query, which can be up to 50), into smaller Objects. We sent every field of the Object as a separate object and tested the system again. The performance of the system improved slightly, but we believe that the reason for so little improvement is the fact that the Image fields are still big objects. We then decided not to send the image as a full Object, but to send it as a list of bytes, in this way the probability of having errors during the communication should be decreased, as the size of the Object sent is much smaller. Finally, we tested the system again, but with much better results.

5.2.5 Stream Concurrent Access Problem

Another problem we meet during testing was that the Mobile Client crashed every time a Central Client sent a message or a result to the Mobile Client, while still receiving data from its last transaction. The problem was that the "send" methods of the ServerClientCommunication interface where concurrently called by different Threads. Streams in Java™ are blocking, but that counts only for the single object sent, by calling the methods of the ServerClientCommunication interface concurrently it happened that some objects mixed with object that were part of another communication. To solve that problem we made all the "send" methods of the ServerClientCommunication interface blocking, in this way, only one thread at the time can send data to the client. Implementing the interface implementation class as a monitor, done in Java™ using the "synchronized" keyword, has done the blocking functionality.

```
public synchronized void sendResult(Object result,
                                    Object DBID,
                                    Object output_connection
                                                ) throws Exception{
      .
      .
}//end sendResult()
```

6. Test Set: Test Document

6.1 Introduction

6.1.1 Scope

This Test Document focuses on the Messaging System, a subsystem of the Car Tracking System.

6.2 Requirements for Test

The listing below identifies those items that have been targeted for testing.

Data Testing

Verify that correct data is retrieved from the database

Functional Testing

The Clients should be able to send queries to the Database and receive results

The Central Client should be able to send a Query Result to the Mobile Client

The system should be able to communicate with a general DBMS

User Interface Testing

The GUIs of the system should be flexible and change automatically depending on the Database structure

Performance Testing

Verify response time

Configuration Testing

Verify the system works correctly in real life configuration

6.3 Test Strategy

This section presents the different kind of tests that we have planned to implement and the way they will be run.

6.3.1 Testing Types

Data and Database Integrity Testing

Verify that correct data is retrieved from the database

Test Objective:	Ensure that the result of the queries sent to the Database is consistent with the data in the database
Technique:	• Send queries from a Client application • Send the same query manually to the DBMS • Check that the two result are equal
Completion Criteria:	Result data from the query is consistent with the data in the DBMS
Special Considerations:	All functional test should be successful before running this test

Function Testing

The Clients should be able to send query to Database and receive results

Test Objective:	Ensure that the Client can send the query and receive the result
Technique:	Execute "User Searches" use-case, using valid and invalid data, to verify the following: • The expected results occur when valid data is used. • The appropriate error or warning messages are displayed when invalid data is used.
Completion Criteria:	• All planned tests have been executed. • All identified defects have been corrected or documented.
Special Considerations:	The system does not accept graphic fields as querying criteria

The Central Client should be able to send a result without request to the Mobile Client

Test Objective:	Ensure that the Central Client can send a specific Query Result to the correct chosen Mobile Client

Technique:	Execute “Central Client Sends Query with no Request” use-case, using valid and invalid data, to verify the following: • The Correct results appear on the Mobile Client when valid data is used. • The appropriate error or warning messages are displayed when invalid data is used.
Completion Criteria:	• All planned tests have been executed. • All identified defects have been corrected or documented.
Special Considerations:	None

The system should be able to communicate with a general DBMS

Test Objective:	Ensure that the system can connect and communicate with a general DBMS.
Technique:	Connect different DBMS to the system and verify the following: • The system successfully connects to the DBMS • The system can send queries to the Database and receive results
Completion Criteria:	• All planned tests have been executed. • All identified defects have been corrected or documented.
Special Considerations:	None

User Interface Testing

The GUIs of the system should be flexible and change automatically depending on the Database structure

Test Objective:	Verify the following: • The GUI represent correctly the information contained in the DB Structure • All navigation and window behaviours are correct
Technique:	• Use different DB structure and check that the system responds with a correct GUI • Test all windows behaviour using an external person to test the GUI
Completion Criteria:	Each window has been successfully verified.
Special Considerations:	The GUI does not accept a graphic field as a querying criteria

Performance Profiling

Verify response time

Test Objective:	Verify that system response time in not unacceptable (5 min)
Technique:	Set system in real life configuration and calculate performance

Completion Criteria:	• The performance of all communication functionalities has been calculated • The response time has been reduced as much as possible
Special Considerations:	GPRS connection in not stable

Configuration Testing

Verify the system works correctly in real life configuration

Test Objective:	Verify that the System responds correctly when set with the real life configuration
Technique:	Run all functional test in real life configuration
Completion Criteria:	• All functional test has been run • All functional test has the same response as in development configuration
Special Considerations:	None

6.4 Test Result

At the release of the Query System, all the needed JUnit™ test cases and manual tests to fulfil the requirements mentioned in S*ection 6.2* were running correctly. A problem occurred regarding the Verify response time, *Section 6.3.1*, but a solution was found for that, see *Section 5.2.4*. Other minor defects occurred but were fixed on the fly.

Messaging System

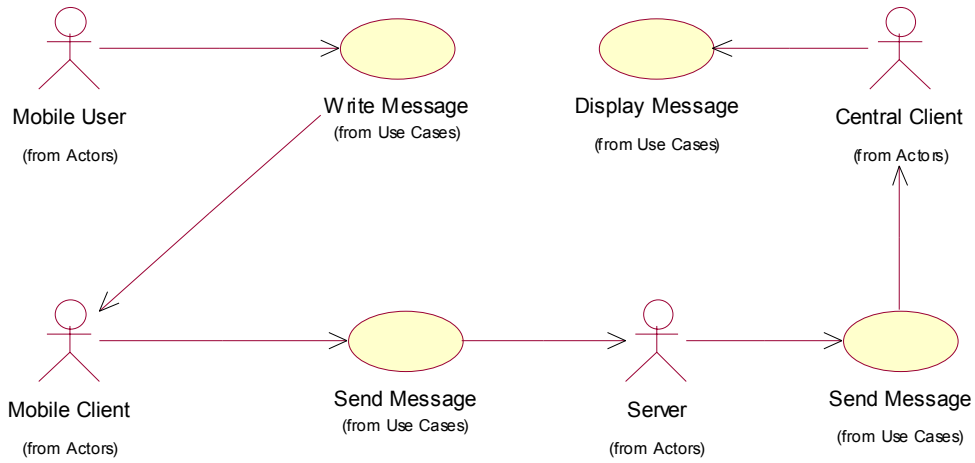
1. Introduction

The Messaging System, the second subsystem of the Car Tracking System, allows the Mobile Users to communicate with the Central Operator. Mobile Users from the Mobile Client application are able to send a message to the responsible Central Operator through the use of a GUI, which allows showing all, sent and received, messages. The Central Operator, differently then the Mobile Users, has the possibility to choose the destination of the message. The Mobile User, in addiction, can also send messages related to query results, which in the following section will be referred as "Action Results".

The chapter starts by describing the most essential use case for the subsystem, other use cases can be found in Appendix C. Thereafter we go in to details of the architecture of the subsystem in the Software Architecture Document, following by the Implementation and the Test documents. Software requirements specifications for the subsystem have been captured and described in the *Software Requirements Specification in the Requirements Set*[1].

2. Requirements Set: Use Case Specification

2.1 Mobile User Message Sending



2.1.1 Details

Description
The Mobile User has to send a message to the Central Client. He/she writes the message and sends it through the Mobile Client. The Server receives the message and dispatches it to the Central Client.
Basic Flow
• The Mobile User selects the messaging text area • The Mobile User writes a message • The Mobile Client sends the message to the Server • The Server receives the message • The Server sends the message to the Central Client

[1] Page 64

• The Central Client receives the message • The Central Client displays the message • The use-case ends
Alternative Flows
1. *Server is Down* An error message is displayed, asking to try again later. 2. *Central Client in not on-line* An error message is displayed at the Mobile Client side, acknowledging the failure of sending the message
Pre – Conditions
The Mobile User is logged on
Post – Conditions
A message is received and displayed at the Central Client
Special Requirements
None
Frequency
More than 10 time a day
Primary Actor
Mobile User
Secondary Actor
Server, Central Client
Secondary Use-Cases
None

3. Requirements Set: Software Requirement Specification

See *Requirement Set; Software Requirements Specification*[1].

4. Analysis & Design Set: Software Architecture Document

4.1 Introduction

4.1.1 Scope

The scope of this Software Architecture Document is to give different views of the Messaging System, (a subsystem of the Car Tracking System), with enough details to give the programmer all the information needed to start the system implementation.

4.2 Architectural Representation

This document represents the architecture of the Messaging System in the form of Use-Case view, Logical view, Deployment view and Implementation view.

[1] Page 64

4.3 Use-Case View

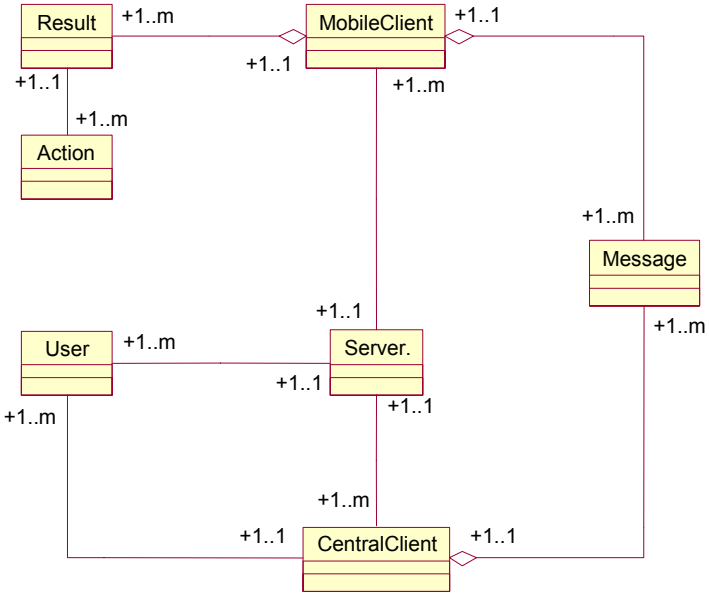
4.3.1 Use-Case Realisations

The Use-Cases realisation can be found in *Appendix C*.

4.4 Logical Views

4.4.1 Analysis Model

Here we have the Logical view of the system but again we leave all details to the design of each of the component.

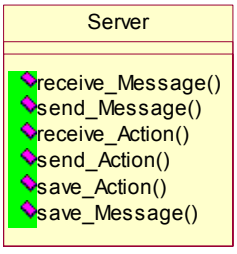


Most of the classes and architecture of this system are similar if not the same as the ones from the Query System. The Server class represent the server that works as a dispatcher of messages between the Mobile Client and the Central Client. All information related to a message is held by the Message class, which is then used by the Server to dispatch the message to the correct receiver. Both the Server and the CentralClient class use the User class in order to identify specific MobileClient those that are online.

Of particular interest is the use of the Result class from the Query System. In this system, the Mobile User must be able to send the result of an action related to a Query Result received from a query. The reason why we decided to put this functionality under the messaging system, is that we treat this action as a message, which is simply related to a Result object.

4.4.2 Design Models

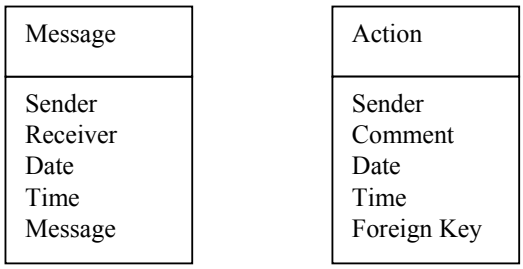
Server Component



As previously explained, the Server class is used as a dispatcher of messages between the Mobile Client and the Central Client. Its main function is to receive messages from one of the clients, select the correct transmission stream, and forward the message to the receiver client. The same goes for Action Results sends by the Mobile Client. Besides the dispatching of Action Results and messages, the Server has the job of keeping a log of all the messages and Action Results sent through the system. The log information will be kept in a database that can be resided locally, in a remote computer, or in the same Database Server used for the Query System.

Data Model

The diagram below shows the data-model to be applied to the database. This data-model concerns the Message and Action classes.



Those two diagrams represent two tables that should be created in the database in order to store the log information. However, here a problem rises, a decision has to be made, in which database those information should be stored. Both tables have a reference to the database used in the administration component (Sender, Receive), which might suggest that those tables should be placed in the same database. However, only the Message table can be placed there, as the Action table has a reference to a Result from a specific database. This should be done using a Foreign Key that represents a way to identify the object the result is related to. For that reason this discussion is considered to be outside the scope of this project, and should be addressed in the future Analysis & Design of the Administration Component.

Functions

N.	Function	Type	Specification	Complexity	Uncertainty
1	Server.receive_Message()	S	N	1	1
2	Server.send_Message()	S	N	1	1
3	Server.receive_Action()	S	N	1	1
4	Server.send_Action()	S	N	1	1
5	Server.save_Action()	U	S	1	1
6	Server.same_Message()	U	S	1	1

Legend

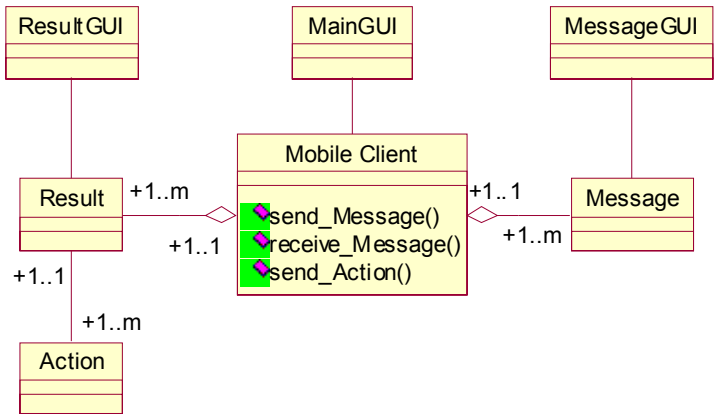
Type	S = Signal
	U = Update
	C = Computing

Specification	N = Name I = Implicit S = Specification
Complexity	1 – 5
Uncertainty	1 – 5

Function Specification

See *Appendix D.*

Mobile Client Component



The MobileClient class is used to send the messages to the CentralClient. As these Clients and the Query System are part of the same system, we decided to integrate those two systems by using the same MobileClient, and just add functionalities to it. For the Query System MobileClient class, there has already been drawn a State Chart diagram, and this diagram, is in our opinion, enough to give an idea for the states of the MobileClient class therefore we decided not to use a new diagram for the Messaging System.

Functions

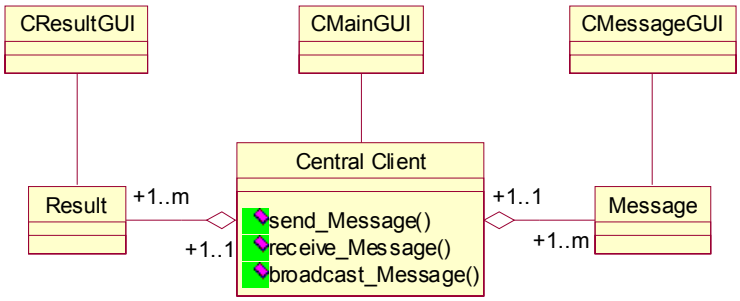
N.	Function	Type	Specification	Complexity	Uncertainty
1	MobileClient.send_Message()	S	N	1	1
2	MobileClient.receive_Message()	S	N	1	1
3	MobileClient.send_Action()	S	N	1	1

Legend

Type	S = Signal U = Update C = Computing

Specification	N = Name I = Implicit S = Specification
Complexity	1 – 5
Uncertainty	1 – 5

Central Client Component



The Central Client Component does not differ too much from the Mobile Client Component; it uses the same classes and has the same functionalities. The only real differences are the functionalities for the CentralClient class. Differently than the MobileClient class, the CentralClient class has the possibility of choosing the receiver of the message, and it can as well broadcast the message to all the Mobile Clients online.

Functions

N.	Function	Type	Specification	Complexity	Uncertainty
1	CentralClient.send_Message()	S	N	1	1
2	CentralClient.receive_Message()	S	N	1	1
3	CentralClient.broadcast_Message()	S	N	1	1

Legend

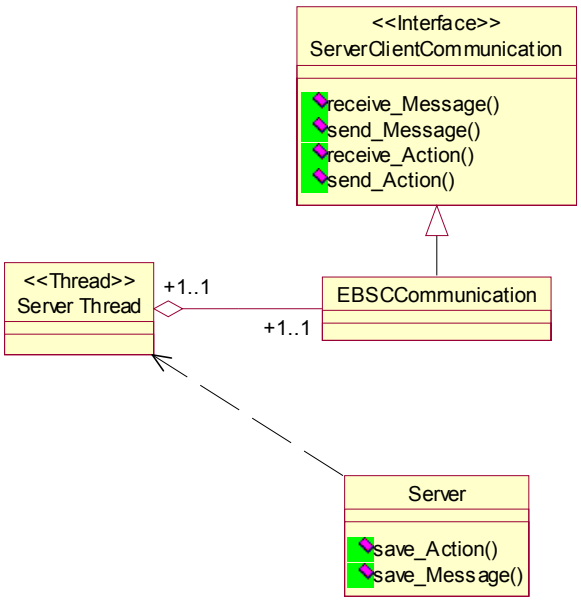
Type	S = Signal U = Update C = Computing
Specification	N = Name I = Implicit S = Specification
Complexity	1 – 5
Uncertainty	1 – 5

4.5 Deployment View

See *Analysis & Design Set; Software Architecture Document.*

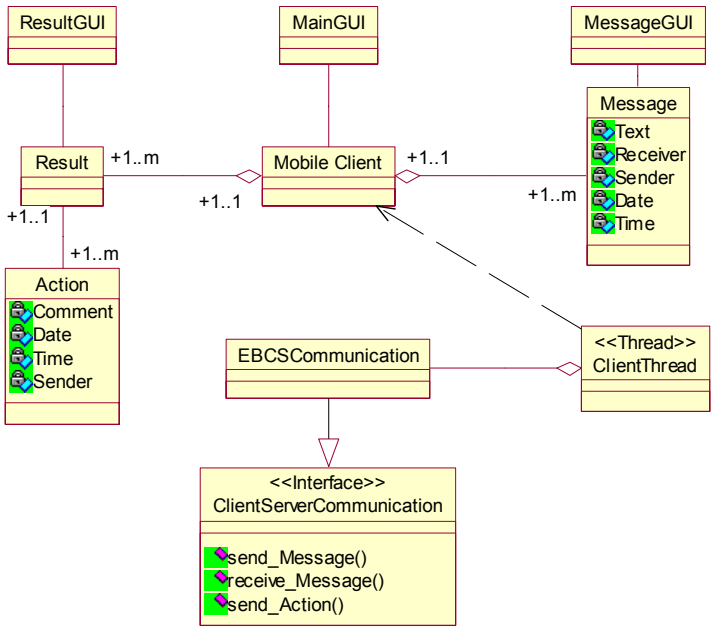
4.6 Implementation Views

4.6.1 Server Component



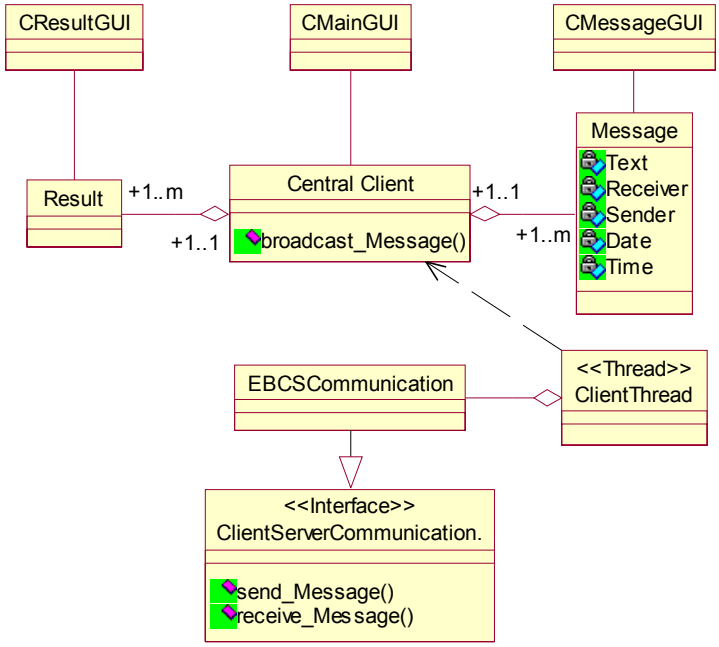
As the Server must be able to handle multiple connections, a Thread run by the Server handles each connection. For system integration reason, we decided to use the same Thread class used in the Query System, by adding to it more functionality. To keep the flexibility of the system all the Messaging System functionality have been moved to the Interface implementing class, in the same way we have done for the Query System.

4.6.2 Mobile Client Component



In order to let the users interact with the system while messages are sent or received, we decide to create a separate Thread that would handle the communication. To keep the flexibility of the system all communication functionality have been transferred to the ClientServerCommunication interface. In the same way as we did for the Server Component, we reused the classes from the Query System.

4.6.3 Central Client Component



The only difference for the Central Client Component is the “broadcast_Message()” function that has not been moved to the ClientServerCommunication Interface for performance reason, as more detailed description of this decision is documented in the *Implementation Section*,of this system.

5. Implementation Set: Implementation Document

5.1 Introduction

5.1.1 Scope

Features implemented for this system are the sending of messages between Mobile Client and Central Client, and sending Action Results form the Mobile Client to the Central Client.

5.2 Implementation Issues

5.2.1 Message Broadcast

As previously mentioned in the Software Architecture Document of this system, the “broadcast_Message()” function of the CentralClient class, has not been moved to the ClientServerCommunication interface as all the other communication function, for performance reason. If we where to broadcast a message from the Central Client we would have to go through all the Online Mobile Clients and send the message to all of them; the message will then have to go through the Server and then finally reach the Mobile Client.

The problem here is that between the Central Client and the Server there is only one communication Stream, which means that the messages will be sent sequentially to the Server and each message will have to wait for the previous one to arrive to the Server before it can be sent. If large number of Mobile Clients were online, this would slow down the connection speed of the Central Client.

Our solution has been to send the message from the central Client to the Server with a special code for the "Receive" attribute of the "Message" object. Every time the Server reads this special code, it will go through all the Mobile Clients connections and send the message. The improvement in performance is because the Server, differently than the Central Client, has a communication Stream for each Mobile Client online, which means that the messages will be sent concurrently.

5.2.2 Send Action Result

As at this stage, things were still unclear about the actual format of the Action Result, we decided to implement it as a message for this demo version. Basically, the Mobile Client creates a message, which contains a textual representation of the Action Result and the related Result object. The Message is then treated as a normal message. Once more detailed information about the Action Result Object is obtained then this should be changed; e.g. an ActionResult class could be created holding all needed information and the functionalities related to it.

5.3 Limitations

5.3.1 Save Messages & Action Results

The messages should be kept in a database that will register sender, receiver, date, time and message sent through the network. As Messages and Action Results are related to Users the discussion of which database should be used to store those information is considered to be outside the scope of this project, and should be addressed in the future Analysis & Design of the Administration Component

6. Test Set: Test Document

6.1 Introduction

6.1.1 Scope

This Test Document focuses on the Messaging System, a subsystem of the Car Tracking System.

6.2 Requirements for Test

The listing below identifies those that have been selected as targets for testing. This list represents what will be tested.

Functional Testing

The Mobile Clients should be able to send a message to the Central Client

The Mobile Client should be able to send an Action result to the Central Client

The Central Client should be able to send a message to a specific Mobile Client

The Central Client should be able to broadcast a message to all Mobile Clients

Configuration Testing

Verify the system works correctly in real life configuration

6.3 Test Strategy

This section presents the different kind of tests that we have planned to implement and the way they will be run.

6.3.1 Testing Types

Function Testing

The Mobile Clients should be able to send a message to the Central Client

Test Objective:	The Mobile Clients should be able to send a message to the Central Client
Technique:	Execute "Mobile User message sending" use-case, using valid and invalid data, to verify the following: • The expected results occur when valid data is used. • The appropriate error or warning messages are displayed when invalid data is used.
Completion Criteria:	• All planned tests have been executed. • All identified defects have been corrected or documented.
Special Considerations:	On the demo version the message is broadcasted to all Central Clients

The Mobile Client should be able to send an Action Result to the Central Client

Test Objective:	Ensure that the Mobile Client can send an Action Result to the Central Client and that the Result Object associated to it is the correct one
Technique:	Execute "Mobile User Action Result Sending " use-case, using valid and invalid data, to verify the following: • The correct Action Result appears on the Central Client when valid data is used. • The appropriate error or warning messages are displayed when invalid data is used.
Completion Criteria:	• All planned tests have been executed. • All identified defects have been corrected or documented.
Special Considerations:	The Result Object associated to the action is referred by the first field of the Result (in the demo version)

The Central Client should be able to send a message to a specific Mobile Client

Test Objective:	Ensure that the Central Client can send a Message to a specific Mobile Client
Technique:	Execute "Central Operator Message Sending " use-case, using valid and invalid data, to verify the following: • The correct message appears on the correct Mobile Client when valid data is used. • The appropriate error or warning messages are displayed when invalid data is used.
Completion Criteria:	• All planned tests have been executed. • All identified defects have been corrected or documented.
Special Considerations:	None

The Central Client should be able to broadcast a message to all Mobile Clients

Test Objective:	Ensure that the Central Client can broadcast a Message to all Mobile Clients
Technique:	Execute "Central Operator Message Sending " use-case, choosing broadcast when choosing destination, to verify the following: • The correct message appears on the correct Mobile Client when valid data is used. • The appropriate error or warning messages are displayed when invalid data is used.
Completion Criteria:	• All planned tests have been executed. • All identified defects have been corrected or documented.
Special Considerations:	None

Configuration Testing

Verify the system works correctly in real life configuration

Test Objective:	Verify that the System responds correctly when set with the real life configuration
Technique:	Run all functional test in real life configuration
Completion Criteria:	• All functional test has been run • All functional test has the same response as in development configuration
Special Considerations:	None

6.4 Test Result

At the release of the Messaging System, all needed JUnit™ test cases and manual tests to fulfil the requirements mentioned in *Section 6.2* were running correctly. Some minor defects came up but were fixed on the fly.

Tracking System

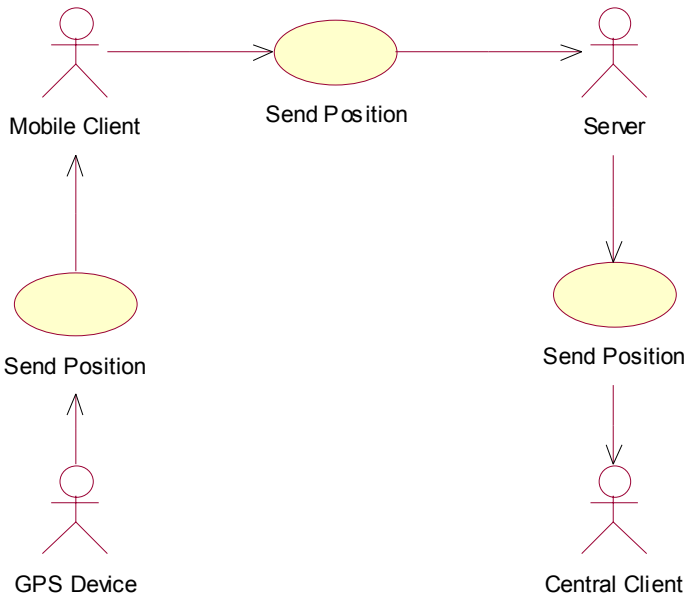
1. Introduction

The third subsystem of Car Tracking System is the Tracking System. This system is designed to allow the Central Operator to have an overview over the vehicle fleet of the company. Every vehicle is equipped with a GPS device that, through the system, sends its position to the Central Operator, which is then able to visualise the vehicles using a GIS. Moreover, the system allows specific objects to be displayed on the GIS, such as buildings, and have those objects sent from the Central Operator to the Mobile Users, or received through queries to a database.

The chapter starts by describing the most essential use case for the subsystem, other use cases can be found in Appendix C. Thereafter we go in to details of the architecture of the subsystem in the Software Architecture Document, following by the Implementation and the Test documents. Software requirements specifications for the subsystem have been captured and described in the *Software Requirements Specification in the Requirements Set*[1].

2. Requirements Set: Use Case Specifications

2.1 Mobile Client Sends Position To Central Client



2.1.1 Details

Description
The Mobile Client receives the position from the GPS device and sends the position to the Central Client through the Server

[1] Page 64

Basic Flow
• The GPS Device sends the position to the Mobile Client • The Mobile Client receives the position and send it to the Server • The Server receives the position • The Server sends the position to the Central Client • The Central Client receives the position • End of the use-case
Alternative Flows
1. Server is Down *An error message is displayed, asking to try again later.*
Pre – Conditions
None
Post – Conditions
The Mobile Client position is received by the Central Client
Special Requirements
None
Frequency
Every 2 seconds
Primary Actor
Mobile Client
Secondary Actor
Server, GPS Device, Central Client
Secondary Use-Cases
Display Position

3. Requirements Set: Software Requirement Specification

See *Requirement Set; Software Requirements Specification*[1].

4. Analysis & Design Set: Software Architecture Document

4.1 Introduction

4.1.1 Scope

The scope of the Software Architecture Document is to give different views of the Tracking System, (a subsystem of the Car Tracking System), with enough details to give the programmer all the information needed to start the system implementation.

4.2 Architectural Representation

The architecture of the Tracking System is represented in the form of Use-Case views, Logical views, Deployment views and Implementation views.

[1] Page 64

4.3 Use-Case View

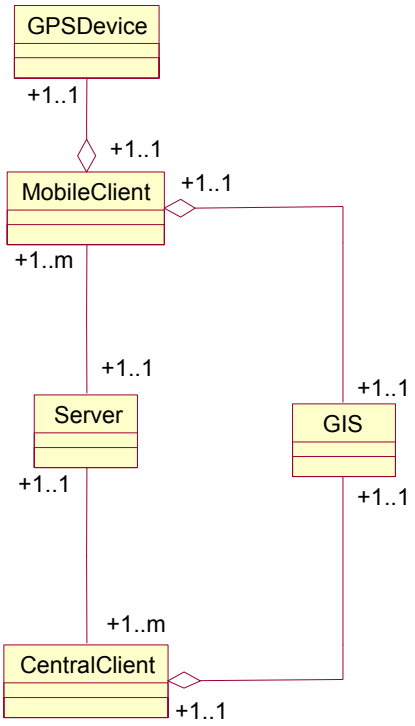
4.3.1 Use-Case Realisations

The Use-Cases realisation can be found in *Appendix C*.

4.4 Logical Views

4.4.1 Analysis Model

The following diagram shows the Logical view of the system; it does not have any details because this is left to the design of each component. Following the diagram, we will give a brief description of the overall system, explaining in a general manner the classes and the connection between them.

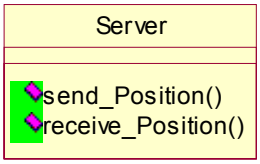


The Tracking System makes use of the same core classes (MobileClient, CentraClient and Server) as the two previous described systems. However, differently then the other two, this system interfaces with a GIS system, which means that it is, basically, just a bridge between the Car Tracking System and the GIS system in use. Information for the GIS is exchanged between the clients through the Server in the same way it has been done for the other two systems.

Information about the position of the Mobile Client is received by a GPS device, which is represented by the GPSDevice class. The position of the Mobile Client is then showed in the local GIS of the Mobile Client and then successively sent through the network to the Central Client, which will then be able to track the locations of all the Mobile Clients online.

4.4.2 *Design Models*

Server Component



The Server Component is used as an information exchange mean, also in this system. The position of the Mobile Clients is received by the Server and then sent to the Central Clients. At this stage, this is the only functionality that has been associated to the Server Component.

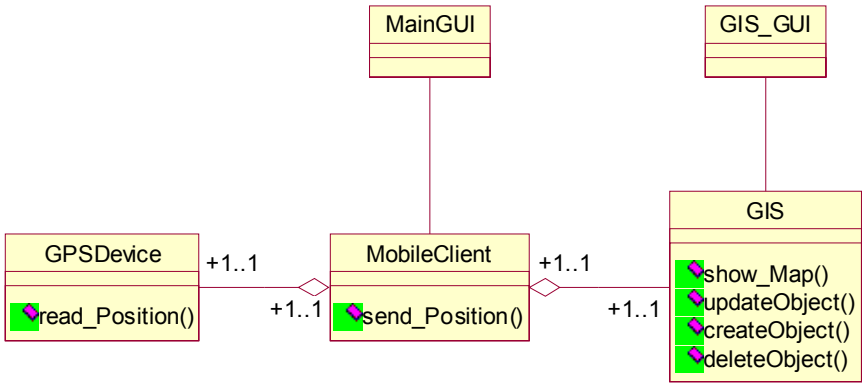
Functions

N.	Function	Type	Specification	Complexity	Uncertainty
1	Server.receive_Position()	S	N	1	1
2	Server.send_Position()	S	N	1	1

Legend

Type	S = Signal U = Update C = Computing
Specification	N = Name I = Implicit S = Specification
Complexity	1 – 5
Uncertainty	1 – 5

Mobile Client Component



The MobileClient is one of the two clients of this system and it is used to send the actual position of the Mobile User to a GIS, which then displays them in a graphical manner. The GPSDevice class represents the GPS device that sends the position of the vehicle to the Mobile Client. The GPSDevice class is used to handle all communication between the Mobile Client application and the actual GPS device.

The GIS class represents the GIS software, which the Car Tracking System interfaces with, and is merely a bridge between the two systems. The only real GUI of this component is the GIS_GUI, which will show the map coming from the GIS software. The Main_GUI has been added to this diagram only to symbolise the integration point with the Car Tracking System's GUIs.

Functions

N.	Function	Type	Specification	Complexity	Uncertainty
1	MobileClient.send_Position()	S	N	1	1
2	GPSDevice.read_Position()	R	S	?	?
3	GIS.show_Map()	S	S	?	?
4	GIS.updateObject()	U	S	1	1
5	GIS.createObject()	U	S	1	1
6	GIS.deleteObject()	U	S	1	1

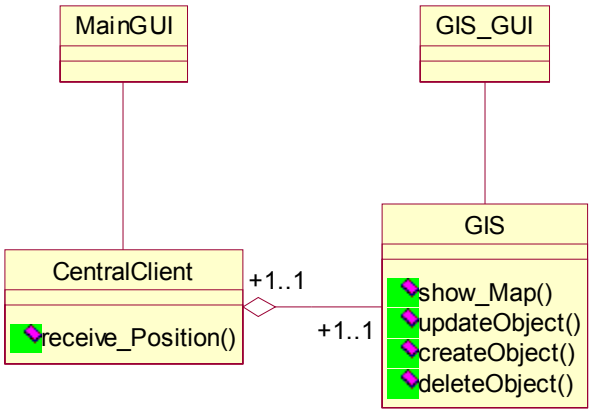
Legend

Type	S = Signal U = Update C = Computing
Specification	N = Name I = Implicit S = Specification
Complexity	1 – 5
Uncertainty	1 – 5

Function Specification

See *Appendix D*.

Central Client Component



The Central Client component has a connection to the GIS system in the same way as the Mobile Client, with the difference that the Central Client does not receive the position from a GPS device; instead, it receives the position of all Mobile Users online from the Mobile Clients themselves and displays it on the GIS.

The GIS functions are not mentioned here, as they do not change between this and the Mobile Client component.

Functions

N.	Function	Type	Specification	Complexity	Uncertainty
1	CentralClient.receive_Position()	S	N	1	1

Legend

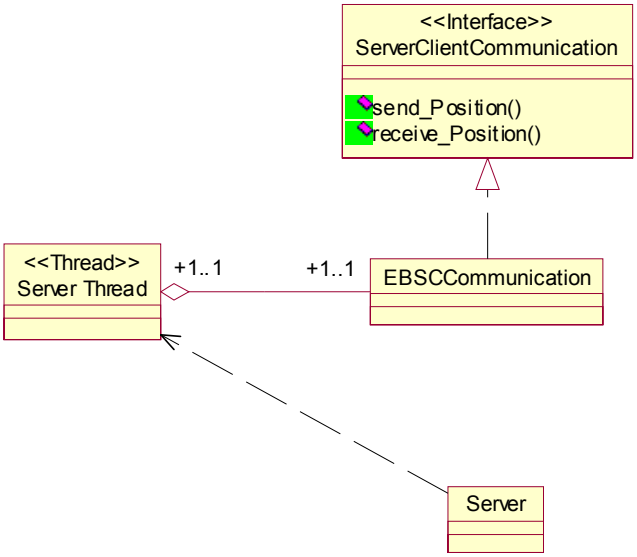
Type	S = Signal U = Update C = Computing
Specification	N = Name I = Implicit S = Specification
Complexity	1 – 5
Uncertainty	1 – 5

4.5 Deployment View

See *Analysis & Design Set; Software Architecture Document.*

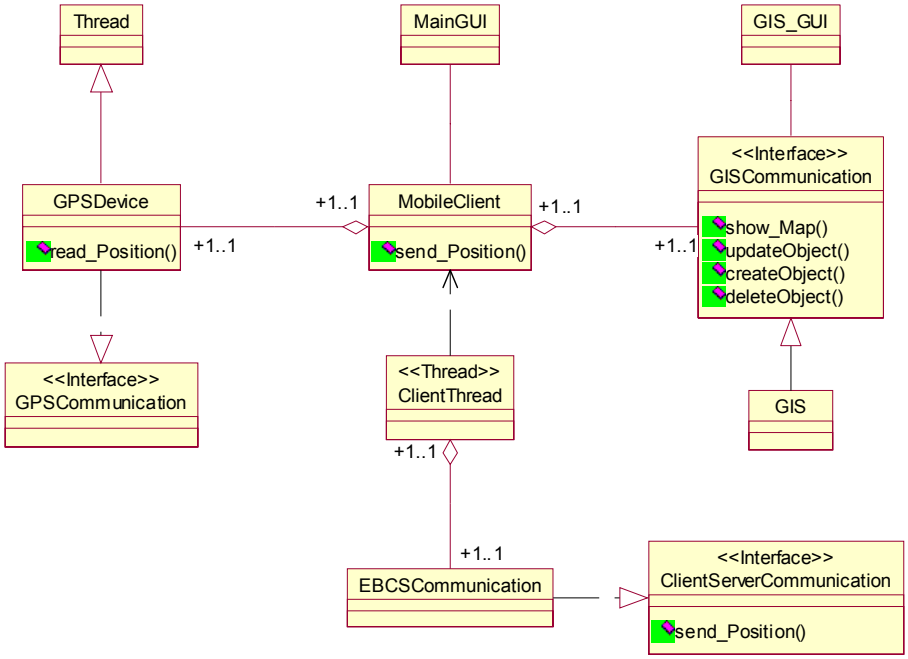
4.6 Implementation Views

4.6.1 Server Component



As for the other two systems, we run a thread form the Server in order to handle multiple connections. To integrate this system to the Car Tracking System, we use the same class as the other two systems and just add the needed functionalities. To keep the flexibility of the system all communication functionality have been moved to the ServerClientCommunication Interface, so in case of changes in the communication means, then only the EBSCCommunication class needs to be changed, without affecting the rest of the system.

4.6.2 Mobile Client Component

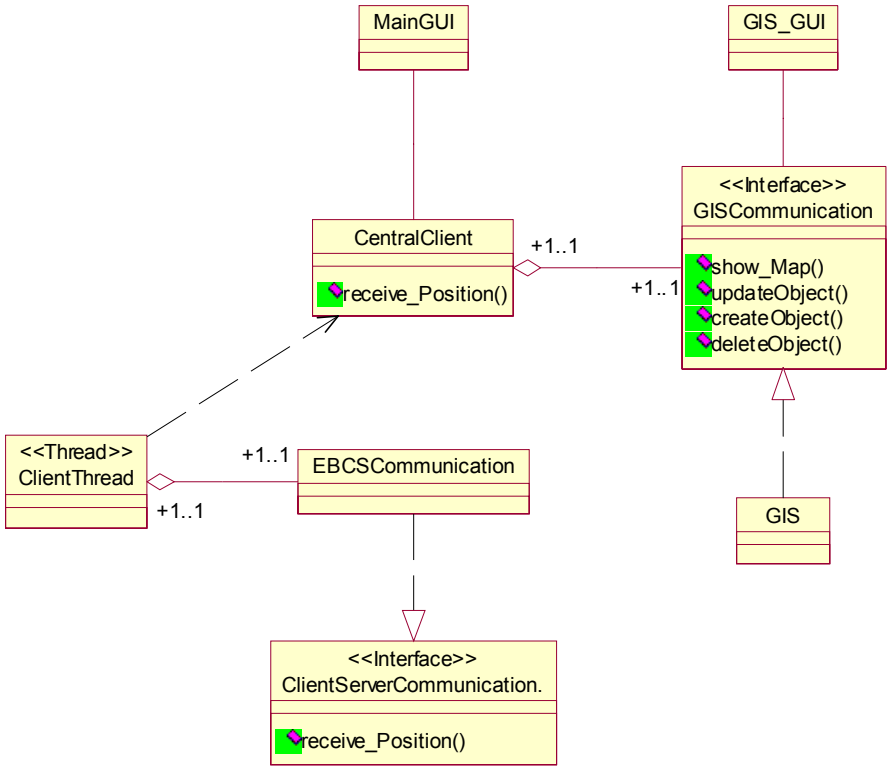


In this component the communication with the Server is the same as for the other Messaging and Query System; a thread is run by the Mobile Client handling the communication with the Server, and all functionality are moved to the ClientServerCommunication interface to keep the flexibility of the system. However, of interest in this diagram is the decision of creating an interface for the communication with the GIS System. We decided to implement it in this way in order to leave the chance of easily changing the interfaced GIS System, for future development of the System.

The GPSDevice class has been implemented as a thread so that all the processing handling the communication with the GPS device will be run by a different process then the main one, in order not to slow down the Clients and to keep the interaction User/Client as smooth as possible. In other words, if the main process handled the communication, then every time the GPS device sends the position to the application, the main process will have to stop the current operation to handle the GPS communication.

Furthermore, we decided to create an interface for the GPS Device in order to give a guidance of the functions used within the system. In this way, if the company decides to change the GPS device, the developers will have a guidance of how the system interacts with the device.

4.6.3 Central Client Component



All implementation comments made for the Mobile Client component apply also to the Central Client component; therefore, we decided only to show the implementation diagram as the component documentation.

5. Implementation Set: Implementation Document

5.1 Introduction

5.1.1 Scope

It is important to realise the scope of the demo version of the Tracking System. The demo is implemented so that is can show the main and most important features of the system. Those features are the display of the geographical position of a Mobile User using the GIS on the Mobile Client side, and the display of the geographical position of all Mobile Users using the GIS on the Central Client side. Furthermore, the Central Client is able to send and display the position of an Object, again using the GIS on the Mobile Client side.

5.2 Implementation Issues

5.2.1 Comment

During the implementation of this subsystem, there have not been any implementation decisions of big importance, because the whole system is basically an interface to another system and the interaction between the systems is already set. However, of more interest are the restrictions of this demo version.

5.3 Limitations

5.3.1 GPS Communication

During the construction of this system we did not had the actual GPS device available. Therefore, we did not have the chance to implement any real interface with the device, but in order not to stop the implementation of the system, we used a simulation algorithm previously used by the Sidabrinis Tinklas to test the GIS. The algorithm feeds the Mobile Client with geographical positions that give the illusion of a moving object.

5.3.2 GIS Object Display

At this stage we did not had enough information from the User Company in order to decide how those Object will have to be exchanged and displayed in the GIS. Our guess is that the Objects are related to the results coming from a query to the database. As the GIS system is able to run searches based on street names (which we think is what the User Company will use), we decided not to implement this feature at this moment and just show the how the function works on the GIS system, manually, if the User Company finds it necessary.

5.3.3 GIS Integration

One of the requirements set by the User Company is that the Car Tracking System should run on a single window, which means that the GIS system should be fully integrated in the Car Tracking System. In order to integrate the GIS System in our system we need to use some kind of component that can be used by our system.

The Akis[1] GIS system development team has developed an ActiveX® component for the GIS System, mainly designed for Internet applications. This component could be used by our system in order to integrate the GIS, but Java™ does not support ActiveX® components. A solution will though be to create a bridge between Java™ and the specific ActiveX® component using C++ and Java™ Native methods.

However, as this solution will take a long time to be implemented we decide not to integrate the GIS system for the demo version, and just keep it as a self-standing application with which our system communicates.

6. Test Set: Test Document

6.1 Introduction

6.1.1 Scope

This Test Document focuses on the Tracking System, a subsystem of the Car Tracking System.

6.2 Requirements for Test

The listing below identifies those that have been selected as targets for testing. This list represents what will be tested.

Data Testing

Verify that correct data is stored in the GIS System databases

Functional Testing

The Mobile Client should be able to display its geographical position in the GIS

The Central Client should be able to display all Mobile Clients geographical position in the GIS

[1] Akis© V.Paliulionis

Configuration Testing

Verify the system works correctly in real life configuration

6.3 Test Strategy

This section presents the different kind of tests that we have planned to implement and the way they will be run.

6.3.1 Testing Types

Data and Database Integrity Testing

Verify that correct data is stored in the GIS System databases

Test Objective:	Ensure that the data stored in the GIS databases is consistent and correct
Technique:	Execute all use-cases, to verify the following: • The data read from the GPS device is the same as the one stored in the GIS databases of the Mobile Client • The data stored in the GIS databases in the Central Client is the same as the data from all online Mobile Client's GIS databases
Completion Criteria:	Read data from the GPS is consistent with the data in the DBMS
Special Considerations:	All functional test should be successful before running this test

Function Testing

The Mobile Client should be able to display its geographical position in the GIS

Test Objective:	The Mobile Client should be able to display its geographical position in the GIS
Technique:	Execute "Display Position" use-case, while reading the position from the GPS device, and verify the following: • The Mobile Client position is displayed in the GIS • The Mobile Client geographical position is correct
Completion Criteria:	• All planned tests have been executed. • All identified defects have been corrected or documented.
Special Considerations:	On the demo version, it is not possible to verify the geographical position, because the demo only uses a simulation and not real data from the GPS device.

The Central Client should be able to display all Mobile Clients geographical position in the GIS

Test Objective:	Ensure that the Central Client is able to display all Mobile Clients on the GIS
Technique:	Execute "Mobile Client Sends position to Central Client" use-case, to verify the following: • The Correct Mobile Client is displayed in the GIS • All Online Mobile Clients are displayed in the GIS • All Mobile Clients geographical position is correct
Completion Criteria:	• All planned tests have been executed. • All identified defects have been corrected or documented.
Special Considerations:	On the demo version, it is not possible to verify the geographical position, because the demo only uses a simulation and not real data from the GPS device.

Configuration Testing

Verify the system works correctly in real life configuration

Test Objective:	Verify that the System responds correctly when set with the real life configuration
Technique:	Run all functional test in real life configuration
Completion Criteria:	• All functional test has been run • All functional test has the same response as in development configuration
Special Considerations:	None

6.4 Test Result

At the release of the Tracking System, all needed JUnit™ test cases and manual tests to fulfil the requirements mentioned in *section 6.2* were running correctly. Some minor defects came up but were fixed on the fly.

Deployment Set

1. Introduction

There comes a time where your children leave home, it is never an easy transaction. That is what the Deployment Set covers, handing our baby, in this case the product and the related documents, over to Sidabrinis Tinklas for further development. In order to make this transaction smoother we provided a couple of documents to them.

First the Deployment Document, where we describe what it is exactly we are handing over, both files and documents, and also an installation and configuration guide for the product, and last but not least a list of known errors and problematic features of the system. Then we make suggestions on further development of the system in the Demo Development Proposal document.

2. Deployment Document

See *Appendix I*.

3. Demo Development Proposal

3.1 Introduction

3.1.1 Purpose

This document presents a proposal for future development of the demo version of the Car Tracking System. It is intended to capture and convey the significant changes that we think should be made to the system.

3.1.2 Scope

The scope of this document is the final release, made by The Jacks, of the demo version Car Tracking System.

3.1.3 Overview

We describe the present problematic of the demo version, and based on those we present a proposal of next possible steps of the system development.

3.2 Present Demo problematic

3.2.1 Server Implementation

In the present version of the demo, the server is implemented as a stand-alone application, which means that the user must run the server before the system can be used. This implies that a user must login in on the server machine and stay logged on until the server is shut down.

3.2.2 Connection Handling

Connection failure between client and server, and between server and database is not handled. Connection failures cause the clients to lose connection and log off the system. In the worst cases the clients and the server crash.

Communication Opening

The Clients main class handles the attempt to connection to the Server. This should be refactored, moving the connection opening to the `ClientServerCommunication` interface. In this way the flexibility of the System will improved, as all communications would be handled by the interface. At present time the connection is attempted before the application GUI is loaded, causing the clients to crash if the connection fails.

3.2.3 Connection Establishment

The connection to the ISP, in case of connection without LAN, is not automatically handled by the system. The user has to manually use the remote network connection, and run the system once the connection is established.

3.2.4 GIS Data Source

The connection to the GIS is hard coded in the system and it is using an ODBC bridge to connect to the GIS database. Therefore, the client is bound to ODBC and Microsoft Windows® systems.

3.3 Proposals for Future Development of the Demo

We decide to divide the proposal for future development of the demo into two parts: proposals for guided demonstrations and proposals for self-guided demonstrations.

3.3.1 Proposals for Guided Demonstrations

When we talk about guided demonstrations we, we refer to demonstrations that take place in the Development Environment, where a developer leads the customer through the demonstration and is there to assist at all time.

Improve GUI

As most of the basic functionalities have been discovered, we think it will be good to start focusing on the appearance of the system. As the users are to be novice computer users, and due to the limited working environment, they have, then we think would be wise to shift the focus towards the GUI and try to define the basic GUI design before the system becomes more complex.

GIS Integration to Later Stages

The integration of the GIS to the Car Tracking System requires a number of decisions to be taken and to remain stable; decisions such as which GIS software is to be integrated, and the looks of it in the Car Tracking System. Therefore, we think that the integration could be postponed to a later stage, when the GUI design has been almost stabilised and the GIS software has been chosen.

Use of Realistic Data on the Database

The database used at in this demo, contains random data that is often redundant. To give a better impression to the User Company and to have a more realistic demonstration, the data in the database should be replaced with some more realistic ones (e.g. buildings might be assigned to existing addresses).

Implement Image on Demand

During the development of the demo, we noticed a performance problem when the System transfers large images through the GPRS device. A proposed solution of this problem was the implementation of the Image on demand feature. This feature allows the User to see that an image is connected to a query result and gives the possibility of downloading the picture if needed.

Implement Basic Authentication

It would be a good idea to start implementing the basic features of Authentication of Users. The Client application should also be implemented so that it will switch between Mobile and Central Client depending on the user that logs-in.

Keep Configuration on a Text File

As requirements are still not stable, and more are to come, we think it will be a good idea to keep the configuration of the system on a text file, and implement the administration component of the system in a later stage.

3.3.2 Proposals for Self-Guided Demonstrations

Another way of demonstrating the system would be to hand it over to the user company and allow them to install it in their environment for evaluation. The system has to be very stable, as this kind of demonstration would take place without the continuously assistance of the developer.

Server Implementation

Our idea is to hand the potential user companies the software for the Clients, and to keep the Server running in a server computer in the Development Environment. Every user company will be able to connect to the server via different ports that will be hard coded in the client software. At the developer site an instance of the server will be run for each of the companies using a different port number for each of them. The reason for doing so is so that the user companies do not have to deal with any configuration or installation of the Server, which requires certain knowledge of the system.

However, to make the user companies able to connect to the server at anytime, the server should run as a service and not as an application. To do so, the server application will have to be changed to a server object with methods that can be invocated remotely. There is a way to do so with Java™ by using RMI technology.

Connection Handling

As the demo will be handed to a user company and to be run in their environment, it must be made sure that connection failure are well handled by the system and that eventual system failures are automatically notified to the developer team.

Integrate GIS System

The demo software should contain most of the features and functionality of the system and it should be easy for the company to use. Therefore, the GIS should, at this stage, be integrated to the system.

Database Connection

To which database to connect to, should be left to the user company to decide, it could either be to the Testing database of the developer team or to a database of the company.

Keep Configuration on a Text File

As only the client part of the system is handed to the user company, the configuration can be left on a text file.

4. Conclusion

This was not the most interesting part of the development, but something that had to be done, in order to create a smooth transaction of the system. With out the Deployment Document the future developers would already be a bit lost, cause it explains what artefacts come with the product and how install and configure it.

Then we provided them with our suggestion of the next steps in the development; we are in better position to see that now, as we have been working with the system and with the users, than some new developers that are just coming to the system. However, due to the reason that this product is to be handed into the hand of future developers but not users then we had to modify the documents a lot to make them suit the need their audience.

Epilogue

1. Evaluation

1.1 The Environment

1.1.1 Our Stay In Lithuania

The stay in Lithuania turned out to be one of the greatest experiences we have had. Our kind colleagues at work, and the friendly people we met during our stay there, have made this period in our lives unforgettable. We found Lithuanians, very friendly always willing to help, and open-minded. Though the project did not give us a lot of space for travelling, due to its strict timeframe, we still got the taste of the Lithuanian life. We lived in Karoliniskes, a suburb of Vilnius, were we could see how the real life in Lithuania is and how older and lesser-developed parts of Vilnius look like. On the other side by working in the centre we could see the impact of the growth in their economy, visualising in new buildings and in general reconstruction of the infrastructure of Vilnius, giving us the idea of Lithuania as a young and fast growing country.

The language, especially in the suburbs, became a little problem, but nothing that could not be solved by gestures. The only real problem with the language was at the start, when we had to find a place to stay. Thanks to our friends Ignas and Audrius, former exchange students at Roskilde Business College, we were able to translate the ads, advertising apartments for rent, in the newspapers. Their help was extremely valuable, also when our landlady tried to engage us in conversations.

1.1.2 Working With Sidabrinis Tinklas

Sidabrinis Tinklas offered us far more than we expected. We had the chance to work directly in the company's environment; we had our own desks and our own computers. We had access to all of their hardware and software resources, and furthermore we had access to their library. All our colleagues were open and willing to assist us when we ran into troubles. Sidabrinis Tinklas really gave us the feeling that we were a part of their team, inviting us to meetings and to social events.

1.1.3 Working With The User Company

The user company had almost no direct contact with us during the initial stage of the project. All the communication went through our contact person at Sidabrinis Tinklas by the means of emails and phone calls. Later on in the project, we had some direct contact with them during the demonstrations of the system; there we were able to have a face-to-face conversation with user companies contact person. Though we did not experience any long delays in getting information from them, then we reckon that if the contact with them would have been more direct and frequent then we would have been able to clarify and gather more specific software requirements for the system.

1.1.4 The Task

One of the biggest fears we had, before our contact with Sidabrinis Tinklas, was that the task that they would give us would not fit with all the aspect of a 5th semester project or our personal interests. Fortunately enough, the task we got really sparked our interests and had very few constrains tied to it, leaving many important decisions to us. Perfect for a practical oriented project, which we had in mind.

We are a very practical oriented team but still with a good background and knowledge of the theory behind the practices, allowing us therefore to combine the two sides together in this report in an efficient manner. We though think that a team with a strong theoretical background would have had problems with this project, as it was very important for Sidabrinis Tinklas to get a working demo of the system. Our suggestion, for teams that would like to do their main thesis abroad, in Lithuania or in other foreign country, is to have an early contact with the company they are to work with, and to establish the what is expected of them and what the task is beforehand.

1.1.5 The Teamwork

Beside our expectations, the teamwork went without any major problems. During the process all tasks that were possible to divided between the team members, fell automatically in the hands of the member with the greatest experience in that specific area, related to the task. Hjörtur, who had more experience of theoretical research and documentation, took most of the tasks related to the actual writing of the report (e.g. editing, rephrasing consideration etc.); while Dario, with a more practical experience, took over most practical tasks such as drawing diagrams, and make researches for better implementations.

The team discussed the most critical parts of the system, and once the decision was made, the tasks were again divided between us. Sometimes the blind believe that one's opinion was right, became a problem especially when our opinions were contradicting; but on the other hand we think that the criticism we put on each other's work helped us in the process of making a better product.

1.2 The Process

1.2.1 The Use of RUP®/XP

Tailoring the RUP® framework was not easy job, but we believe that once this was done properly, RUP® perfectly fitted the project on all its phases. RUP® is a very broad framework that almost covers the whole software development life-cycle; it contains a broad number of tools, practices, and techniques that can be used. Trying to produce the entire mentioned artefact in RUP® would be impossible for a small project of our size. However, RUP® comes with good guidelines on how tailor the process and on how to choose the relevant artefacts for a specific project. Furthermore, RUP® provides the users also with many different examples.

Through those examples and the RUP® guidelines, we were able to select a number of relevant artefacts for our project, which all turned out to be used during the process. The only document that turned out to be of little value for us was the Business Case[1], which has not been used since the first draft as there was need for business analysis in this project as we were given a specific task by Sidabrinis Tinklas to solve.

Though RUP® has all those artefacts we still found it necessary to add some new artefacts which we created specifically for this project, and modified others to suit our project, mainly in order to limit the redundancies of information. This shows how no methodology can perfectly fit one project, but that a flexible methodology like RUP® can easily be modelled to do so.

We were able to combine selected practices of XP with RUP® without any problems. RUP® is mostly oriented in the process that has to be followed in building a system, and gives some suggestion on what are the best practices that can be used during the process, but it almost never ties the process to any specific practices. Therefore, it was no problem using the chosen XP practices within the process framework of RUP®. As a matter of fact, few weeks before the end of our project, Rational Software Corporation® the owner of RUP®, introduced a XP plug-in to the RUP® framework for the ease of combining RUP® and XP together.

1.2.2 The iterative process

This was our first experience of using a proper iterative process, if we exclude a very small project on our 3rd semester of only 2 weeks. Initially we felt a little uncomfortable or insecure about designing and implementing one subsystem at a time because we feared that we would have problems integrating them into the final demo, but we were wrong. Once the general architecture of the system was in place, then we encountered no problems doing so. During the first iteration we were stunned by the number of artefacts that we had to produce, but once the first version of them were done then it was only a matter of small additions and changes that had to be done to them in the following iterations. During the iterative process we often had to use the same artefacts repeatedly, these revisits helped us in finding mistakes, finding new requirements and generally in keeping the documents up-to-date.

[1] Page 56

1.2.3 The XP practices

As previously stated, we encountered no problems combining some selected XP practices into the RUP® framework. The XP practices we selected to use were implementation practices, and they fitted perfectly to our project. The continuous integration gave us the possibility of reviewing the system at all times, and it also gave us the ability to always demonstrate the whole demo to the user company instead of maybe only a single subsystem. By doing so, we received much more input from them because they would then concentrate on the whole system, also on parts that they had actually reviewed earlier. Therefore giving us comments on things that sometimes had slipped their attention during previous demonstrations.

Even though it felt a bit strange at first, then we have to say that pair programming was also a success. It kept us focused on producing reusable and quality codes. While the “driver”, the one with the keyboard, concentrated on making a method work, then the other team member would think of possible problems or mistake.

Again, as we used an iterative process, then refactoring came as a natural practice. We went through the implementation of the same classes over and over again, every time trying to remove redundancy and generally trying to improve the code to better fit the ideal architecture.

Test before coding is one of the XP practices that fascinated us the most, but also the one that we were not able to cope with. We found it very hard to think of test cases before knowing what methods that we were actually going to implement. It is though our believe that this handicap could be over come by experience. However, an alternative approach that we think could work well is to have the “passenger”, the one sitting without the keyboard, thinking and writing the test cases for the specific method that the driver is implementing simultaneously.

1.3 The Product

1.3.1 The Demo

After series of changes and a number of proposed solutions, the user company and us finally agreed upon architecture that today is basis of the system. It seems like the chosen architecture fits perfectly to the needs of the system, by allowing the user company keep a certain level of security between the system and their confidential databases, furthermore giving us the facility of changing parts and components of it with great ease.

By having all the configuration functionality within the Server component, we were able to reflect all the configuration changes to all the clients, without the need to go in and make changes on each one of them. The simplicity of the protocol used, and the use of Java™ Stream, allowed us to easily change and manage the flow of information and their content.

Creating a flexible system, was one of the main goals of our project, and in our opinion, has been achieved. The use of programmable interfaces in the communication between the components has made them quiet independent. Therefore, if any changes are made to one of the components, then these changes will not affect the other components, as long as the changes are compatible with the interfaces.

Another thing supporting the flexibility of the system is the use of a text file to describe the structure of the database. This allows the user company to use the system without having to disclose the actual structure of their database to outsiders of their company. Once the system will be finalised, then the user company will only have to give the correct database structure to the system, and it will then automatically adjust to the new database.

The Quality Range described on page 62, describe the quality criteria of the final system, but here we show only the ones that are relevant to the demo version. With the table below, we decided to show how our product fulfils the quality range set at the start of the project.

Criteria	Quality Range	Achievement
Efficiency	The efficiency of the system it is important because the system handles confidential data that should be transferred correctly.	Thanks to the continuous testing process, we can say that the functions used in the demo are correct and efficient.
Testable	Testing is an important part of any system, it will not be possible to ensure any efficiency if the system could not be tested.	Using JUnit™ as testing tool, the system was implemented with the idea of been testable.
Flexibility	Flexibility is one of the most important sides of this system. The system should be able to be configured to different kind DB servers with the least number of changes.	Using the text file based configuration for the database, we have been able to connect the demo to any DBMS of which there exists a JDBC[1] driver for.
Reusability	As previously explained in the Flexibility Quality, this system is very likely to be reused and connect to other DB servers. Therefore, particular attention should be paid at design and implementation phases.	The system has been implemented using Java™ interfaces, which allowed us to create a component-based system where each component is independent.
Portability	Portability seems not to be a very important part of this system, but as the system is to be designed to be flexible, some thought about portability could be made, especially at implementation time.	Using Java™ to implement our system, we automatically fulfilled the portability criteria. Though we decided to test this and we could successfully have the server running on a Linux machine and two clients running on Windows® machines.

Beyond our initial expectations, we were able to implement a working demo of the final system, covering all the three subsystems. The demo served its purpose of presenting the user company the full potential of the final product. After having evaluated the demo, they decided to invest into further development of the system and to buy special hardware (e.g. car computers, GPS devices etc.).

1.3.2 The Report

The objective of our report was to create a document which would in details describe the produced system, and at the same time show the process of that development. As our project was very product oriented we found it easier to describe the product than the process, and at the start, we had some problems because we figure out that none of the process had really been documented. Therefore, we had to undergo a major review of all artefacts, and modified some of them to aid in describing that process.

We found it particularly hard to build the report in a clear and readable way. The structure of it is based upon RUP® and its sets, where each set contains related RUP® documents that we used. The RUP® documents we produced were very specification oriented, and even though we modified them to include some process related discussions, it still was hard to keep a “red thread” going through it. Therefore, we had the idea of having the initial part of the report discussing the general side of the system and then have a more specification oriented second part, where we go deeper into the three subsystems. In this way, the initial part of the report will give an overview of the process and of the general system, while the second part will be more product oriented and show the outcome of the process.

[1] Sun Microsystems

We are very satisfied with the product specification part of the report. The document presents a full Software Requirement Specification of the system, and the Software Architecture Document presents the design of the system, including discussions of important design decisions. Furthermore, the Implementation document presents the reader with a list of important implementation decision and problems that occurred during the building of the system. Finally the Deployment document further specifies the single elements of the system handed to the company, such as list of classes and files and installation notes, there you can also find a proposal for further development of the demo.

The report contains everything a future developer needs to get a good understanding, not only of the demo but also of the whole system. It fulfils our initial goal of producing a system specification of the system for IT professionals, and it also meets the academic need of explaining the process and decisions made through out the project.

2. Conclusion

Here we will answer the questions we asked in the beginning of the report in the *projects Problem Definition*[1]. There are neither right nor wrong answers to these questions; however we try to substantiate our answers with references into our experience with the project. We divided the questions into practical and theoretical questions.

Practical:

- ***Is it possible to use a demo version of a system to attract costumers?***

Well it is clear that this is the case. The product produced during this project is a proof of that, the potential user of the system evaluated the demo version of the Car Tracking System and as stated in *Evaluation*[2] decided to go ahead with the development of the system and has decided to purchase needed hardware for the next phase of the development. Therefore, we can conclude that if done properly then a demo version of a program can, in a very short time, attract potential costumers.

- ***Can a customised system also become a "shrink-wrapped" product?***

Again, the project and its product speak for themselves. We listened to the requirements of the potential user, and all their requirements were documented in the *Software Requirements Specification*[3] and some even implemented in the demo version. However, we never lost the aim of making this system flexible; it is platform independent and can connect to any DBMS that a JDBC[4] driver exists for. So in fact, we have a product that suits both the initial company and other companies with similar needs.

- ***Can two datamatician students go abroad and work with a local company as a part of their main thesis?***

This one is a bit harder to answer, as the success of the main thesis, in our eyes, depends on the grade we get for it, and that we have not received yet, but we fully believe we have done a very good and thorough job. However, looking at it from another point of view, we can say that two datamatician students are able to go abroad and work with a local company in order to produce a good system, as we have stated in above questions.

[1] Page 19

[2] Page 142

[3] Page 64

[4] Sun Microsystems

So, we can conclude that this is doable but is it feasible? We do not believe so. There are just to many factors that can go wrong and we are talking about the main thesis of the studies, a very important final chapter of the education. We had an advantage in the beginning of the project, we both had a lot of international experience before it, we had lived in four different countries each and therefore used to adapt to new circumstances.

We were very lucky with the company we worked for, the project scope suited the timeframe, the teamwork went well etc. etc. However you are in a different country, far away from your teachers and project supervisor, you are ALONE. Some will say that this is not a problem in today's society, with the easy of emails, but it does not help at all in a project like this, it requires deep discussions. There are simply to many things that can go wrong for us to recommend students to go aboard for their main thesis.

- ***How well are we prepared to handle methods/tools/techniques/technologies outside the curriculum of our studies?***

There were couple of things along the way that we had never encountered prior to this project, e.g. RUP®/XP, testing, GPS, GPRS, XML, GIS, etc. etc. However these were no hindrance for the success of the project, sure the studying of new things stole some time that could have been used in developing the system a little further, but due to the broad general knowledge we have from our four prior semesters, we were able to study and put into practice new things quickly and efficiently.

However, a rose is never without a thorn, the one thing that we experienced troubles with, was the use of JUnit™ and the process of designing and writing tests, as we mentioned in our *Test Se[1]t*.

Theoretical:

- ***Does the chosen methodology suit the scope and size of our project?***

Yes and no. If we would have taken RUP®/XP straight out of the box and applied it to this project, we would never have had accomplished what we have today. But by configuring them to our and the projects needs we stand where we are today with a successful product.

The configured version of RUP®/XP, we used, aided us in producing the necessary documents for future developers and in order of producing a quality demo. It also supported the academic part of the project as we could use many of the produced documents here in this report with very little modifications; they in fact form the basis of the report.

- ***How well does the methodology cover the project management of our project?***

Again, the way we configured the methodologies was to suit the scope and size of the project. As we were only two developers in the group and working very closely together we did not need a big overhead for the project management aspect of it.

The produced documents and management tools we used gave us a good overview and control over the project. Furthermore, they provided ST with a good overview over what we were doing and what were our next steps.

- ***How well does the methodology cover the documentation of our project?***

As we stated in the beginning of our report, we diverted from using pure XP on the project on the grounds of the documentation part of it. Combining it with a tailored version of RUP®, proofed to provide us with all necessary documents. It was a hard process deciding which documents were relevant and which were not. It is our believe that we travelled as lightly, concerning the documents, as possible, considering the project.

[1] Page 89

It was very important for us to be able to document the product in a proper way so it would ease the work and understanding of future developers of the system. We believe we have done so, and along the way produced a report with the focus on the product itself but also on the process of getting there. Therefore, we must conclude that the documentation aspect of the methodologies fulfils the need of the project.

- ***Does the methodology cover the whole lifecycle of our project?***

The methodologies we follow are Software Development methodologies, and are not intended to support anything else, just things that are relevant in the process of developing software. They cover, without a doubt, the whole lifecycle of the practical part of the project but when it comes to the academic part it fails. This is due to the nature of the academic part were we go from A, choosing a methodology, to Z, evaluating and concluding on the project. Furthermore, they do not cover the detailed evaluation of the processes as needed for the academic part.

Therefore, the answer is simply no it does not cover the whole lifecycle of our project.

- ***Is it necessary to tailor the chosen methodology to our project and how easy is it to do so?***

As we mentioned earlier we did not follow the chosen methodologies blindly. We did, and in fact, we had to, configure them to our and our projects needs. Especially since, we combined two methodologies that overlap each other.

Doing so was not such an easy task, mainly because of the reason that we had never worked with either of them. However, we managed to go through them and mould them to our likings. Maybe we are wrong in saying that it was not an easy task, rather we should say that it was a time-consuming task, cause we had to go thoroughly through both of methodologies.

- ***Does the methodology give the developed system the necessary support it needs to go into further development after the project finishes?***

Before we can answer this question, we have to speculate what is it that the developed system needs to go into further development? First, of all something that gives an overview over what the system is supposed to do and what problem it is supposed to solve. Then the architecture of the developed system needs to be clear. Furthermore, how and why it is implemented in the way it has been done.

Our combined methodologies covered all this, suggesting the use of a Vision document, a Software Requirements Specification, a Software Architecture document and an Implementation document for future developers. And a lot of other material that aids them in understanding the developed system, and the needs of the whole system. Last but not least, it requires a Deployment document listing what files and documents are relate to the system and how to use the produced system. Therefore, we can say without hesitation that the combined methodologies provide a good environment for future development of the system.

- ***How easy is it to follow the methodology?***

RUP® has very specific guidelines of how to work, what activities and which artefacts to produce. Very easy to follow once configured to your needs. Has great templates for most of the documents suggested by the methodology, very easy to follow and to tailor to your needs. In general an easy methodology to follow.

XP on the other hand is vaguer in its descriptions of activities and artefacts, e.g. they suggest test everything that needs to be tested, but never say what is it that needs to be tested, and they say produce the artefacts you need, but leave you to decide which ones you need. So we found it a little harder to follow XP, than RUP®, in some ways, however the part of XP that we intergraded with RUP®, were maybe easier to follow than some other practices of XP, except the testing part of it as we stated earlier in *Section 1.2.3*.

To conclude we can say that it was generally easy for us to follow the methodologies in the way that we had configured them.

- ***Is the support and tools provided with the methodology adequate?***

XP does not come with any support or tools, all is though not lost because there is a lot of information on the internet about it and couple of books have written about it as well. This turned out to be enough for us and for the few practises we extracted from XP.

RUP® on the other hand has plenty of support and tools available. It is also a commercial commodity; unlike XP then the RUP® framework is sold to its users. It comes with a very detailed electronic description of everything that is related to the framework and users have the ability to use online support and to buy further software tools to use with RUP®.

So the initial question does not really fit in our situation as XP is not provided by anyone, you cannot "get it" anywhere, so to speak. If we look only at the RUP® part then, yes the support and tools provided by RUP® are indeed adequate. However, as we mentioned above about XP, there was no lack of available information about it, adequate for our small project at least.

3. Final Conclusion

MISSION ACOMPLISHED! We managed to produce a demo version of the system that convinced the potential user of the system to go ahead with the production of a full system. Along side, this we also managed to produce a quality academic report with focus on the product specification. Our partners at Sidabrinis Tinklas were very satisfied with the project, the user company expressed their satisfaction with the product, and last but not least we, The Jacks, are extremely satisfied with the final outcome and the over all process.

From this project, we take with us a bundle of knowledge and experience that will come in handy when we enter the "real" world.

Appendices

Table of Contents for the Appendices

Appendices

Appendix A

1. Abbreviations

Abbreviation	Meaning
ASP	Active Server Pages
CSS	Cascading Style Sheets
DB	Database
DBMS	Database Management System
DTD	Document Type Definition
GIS	Geographical Information System
GPS	General Positioning System
GUI	Graphical User Interface
IP	Internet Protocol
LAN	Local Area Network
Msg	Message
PC	Personal Computer
RDB	Rational Database
RDBMS	Rational Database Management System
RUP®	Rational Unified Process
SAD	Software Architecture Document
SQL	Structured Query Language
SRS	Software Requirement Specification
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
XML	eXtensible Mark-up Language
XP	eXtreme Programming
XPath	XML Path Language
XSL	eXtensible Style Language

Appendix B

1. Iteration Plans

1.1 Introduction

1.1.1 Purpose

The purpose of this document is to keep track of the planned activity, and to keep a valuable resource of knowledge for similar projects in the future.

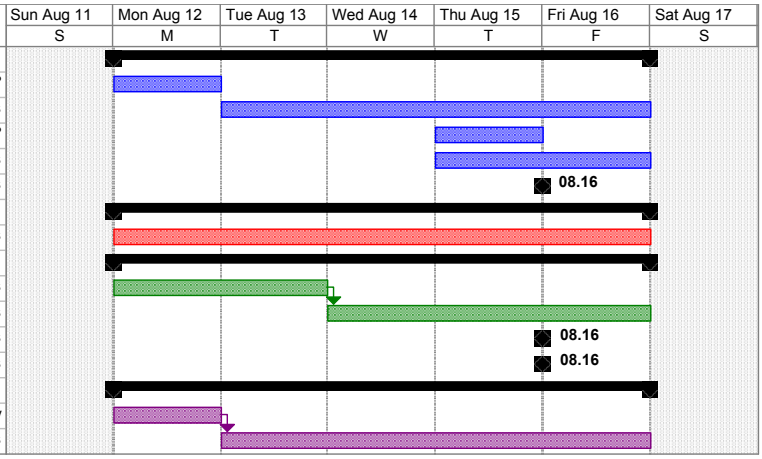
1.2 Plan for Iteration 0

1.2.1 Duration

12th of August – 16th of August

1.2.2 Overview

ID	Name	Duration
1	**Project Management**	**5 days?**
2	Develop Software Development Plan	1 day?
3	Monitor & Control Project	4 days
4	Plan Next Iteration	1 day?
5	Evaluate Scope and Risk	2 days
6	M1 - Establishment	0 days
7	**Requirments**	**5 days**
8	Manage changing Requirment	5 days
9	**Analysis & Design**	**5 days**
10	Performe Architectural Analysis	2 days
11	Perform Architecturel Design	3 days
12	M2 - Architecture Design Document	0 days
13	M3 - Query System Proposal	0 days
14	**Configuratio & Change Management**	**5 days**
15	Prepare configuration plan	1 day
16	Manage Releases	4 days



1.2.3 Deliverable Documents

- Vision Document
- Business Case
- Software Development Plan
- Query System “Solution Options”
- System Architecture Proposal for User Company
- Development Case
- Next Iteration Plan

1.2.4 Deliverable Releases

None

1.2.5 Milestones

Milestone: M1-Establishment	**Date: 16/08/2002**
Description: At this stage, the project should have finished its initial stage. During the stage the following should have been considered: risks and how to avoid them, planning, is this project of interest to the user company, is there a common vision of the system with all involved parties.	
Deliverables: • Risk List Document • Software Development Plan • Iteration Plans • Business Case • Vision Document • Development Case	
Evaluation Criteria: Agreement that the right sets of requirements have been captured and that there is a shared understanding of these requirements. In addition, an agreement that the schedule estimates, priorities, risks, and development process is appropriate. All known risks, at the time, have been identified and a mitigation strategy exists for each, where applying.	

Milestone: M2-Architectural Design	**Date: 16/08/2002**
Description: An overall design of the architecture should be delivered at this point. This document is necessary for the continuation of the project.	
Deliverables: System Architecture Design Document	
Evaluation Criteria: The document should contain the proposed solution for the architectural structure emphasising on how long it will take to add a database, and the time for implementation.	

Milestone: M3- Query System Proposal	**Date: 16/08/2002**
Description: At this stage a document should be released to the User Company to propose the system	
Deliverables: Query System, Solution Options <1.0> for User Company	
Evaluation Criteria: The Architectural design should include the basic architecture of the system. The document should include a UML model of the system, and list all the possible architectures that could be used, the one proposed and a brief explanation of the reason it has been chosen.	

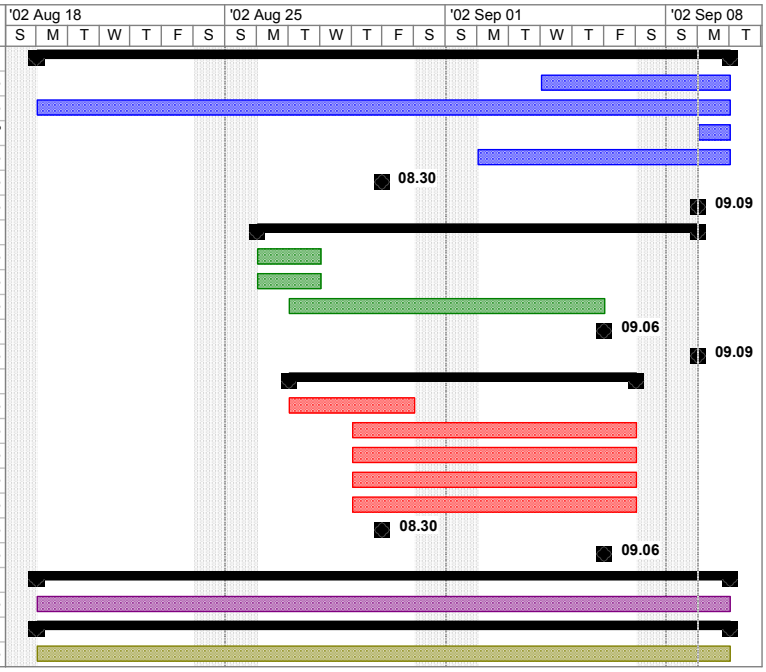
1.3 Plan for Iteration 1

1.3.1 *Duration*

19th of August – 9th of September

1.3.2 *Overview*

ID	Task Name	Duration
1	**Project Management**	**16 days?**
2	Develop Software Architecture Documen	4 days
3	Moditor & Control Project	16 days
4	Plan Next Iteration	1 day?
5	Evaluate Scope & Risks	6 days
6	M4 - Software Architecture Document	0 days
7	M5 - Iteration2 Plan	0 days
8	**Analysis & Design**	**10 days**
9	Find Use Cases	2 days
10	Define Logical View	2 days
11	Define Implementation Model	8 days
12	M9 – Software Requirements Specificatic	0 days
13	M6 - Software Architecture Document	0 days
14	**Implementation**	**9 days**
15	Prepare Test Units	4 days
16	Implement Units	7 days
17	Test Code	7 days
18	Integrate Code	7 days
19	Test System	7 days
20	M7 - Test Units	0 days
21	M8 - DB Interface Units	0 days
22	**Configuration & Change Management**	**16 days**
23	Manage Releases	16 days
24	**Requirments**	**16 days**
25	Manage Requirment Changes	16 days



1.3.3 *Deliverable Documents*

- System Architecture Document
- Test Units
- Iteration 2 Plan
- Software Requirements Specification For Car Tracking System
- Software Requirements Specification For Query System
- Software Requirements Specification For Messaging System

1.3.4 *Deliverable Releases*

Query Interface that allows a mobile client to connect to a central server through GPRS connection and allows him/her to send direct SQL queries and receive results. No user interface at this stage.

1.3.5 *Milestones*

Milestone: M4-Software Architecture Document	**Date: 30/08/2002**
Description: A first release of the Software Architecture Document should be released, in order to give a basis for the construction phase.	

Deliverables:
SAD – Software Architecture Document
Evaluation Criteria:
The Document should include all the use case view and logical view of the system, in the form of UML diagrams.

Milestone: M5 – Iteration 2 Plan	**Date: 09/09/2002**
Description:	
At this stage, a detailed plan for the next iteration should be created.	
Deliverables:	
Iteration Plans Document	
Evaluation Criteria:	
The Iteration Plans Document should be updated and should include a detailed plan for iteration 2.	

Milestone: M6 – Software Architecture Document	**Date: 09/09/2002**
Description:	
At this stage, the Software Architecture Document should be concluded, for the part regarding the Database Interface for the Query System.	
Deliverables:	
Software Architecture Document	
Evaluation Criteria:	
The Software Architecture Document should include use case, logical and implementation view of the Database interface part of the Query System.	

Milestone: M7 – Test Units	**Date: 30/08/2002**
Description:	
Test units should be ready at this stage. Those units will be used to test the code and the system.	
Deliverables:	
Test Units	
Evaluation Criteria:	
A list of test classes made using JUnit™ is ready to be used during testing phases.	

Milestone: M8 – DB Interface Units	**Date: 06/09/2002**
Description:	
DB Interface Units are ready to be integrated in the system	

Deliverables: DB Interface Classes
Evaluation Criteria: Java™ Implemented Units should be ready and well tested. JUnit™ tests should give a 100% correctness result, and so should the integration test.

Milestone: M9 – Software Requirements Specification	**Date: 06/09/2002**
Description: All known requirements of the Car Tracking System should be documented in a SRS document. A more detailed SRS documents should be ready for the Query and Messaging Systems.	
Deliverables: SRS For The Car Tracking System SRS For The Query System SRS For The Messaging System	
Evaluation Criteria: The documents should be approved by our Quality Group	

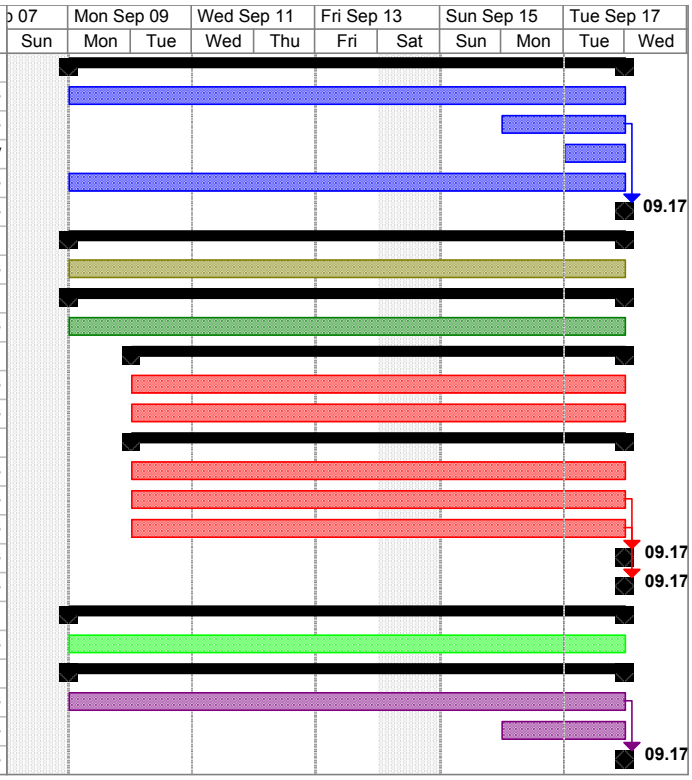
1.4 Plan for Iteration 2

1.4.1 Duration

9th of September – 17th of September

1.4.2 Overview

ID	Task Name	Duration
1	**Project Management**	**8 days**
2	Monitor & Control Project	8 days
3	Plan Next Iteration	2 days
4	Assess Iteration	1 day
5	Re-evaluate Scope & Risks	8 days
6	M10 - Iteration 3 Plan	0 days
7	**Requirements**	**8 days**
8	Manage Changing Requirements	8 days
9	**Analysis & Design**	**8 days**
10	Review SAD	8 days
11	**Implementation**	**7 days**
12	Implement Units	7 days
13	Integrate Code	7 days
14	**Test**	**7 days**
15	Prepare Test Units	7 days
16	Implement Test Units	7 days
17	Test Code	7 days
18	M11 - Test Units	0 days
19	M12 - Query System	0 days
20	**Environment**	**8 days**
21	Support Enviroment	8 days
22	**Configuration & Change Management**	**8 days**
23	Change and update Documents	8 days
24	Manage Releases	2 days
25	M13 - All Documents Updated	0 days



1.4.3 Deliverable Documents

- Test Units
- Iteration 3 Plan

1.4.4 Deliverable Releases

The Query System where the mobile user is able to send queries to the database and get results displayed on the screen. He/she can then look through 20 latest results. The central operator is able to do the same and pass the result to the mobile users if needed. The central operator can also see all connected mobile users.

1.4.5 Milestones

Milestone: M10 – Iteration 3 Plan	**Date: 17/09/2002**
Description: At this stage, a detailed plan for the next iteration should have been created.	
Deliverables: Iteration Plans Document	
Evaluation Criteria: The Iteration Plans Document should be updated and should include a detailed plan for iteration 3.	

Milestone: M11 – Test Units	Date: 17/09/2002
Description: All test units for the Query System should be ready at this stage.	
Deliverables: Test Units	
Evaluation Criteria: Test classes for every unit of the Query System, that can be tested with JUnit™ are ready.	

Milestone: M12 – Query System	Date: 17/09/2002
Description: The Query System with user interfaces, both for the central operator and the mobile user, is ready	
Deliverables: The Query System with all source codes	
Evaluation Criteria: The system has to meet the criteria set in *section 1.4.4* and approved by the Quality Committee.	

Milestone: M13 – All Documents Updated	Date: 17/09/2002
Description: All documents that have been produced during the project are to be updated reflecting a current situation of the project.	
Deliverables: Updated versions of all documents mentioned in all prior Milestones	
Evaluation Criteria: A consistency is kept through out the project documents.	

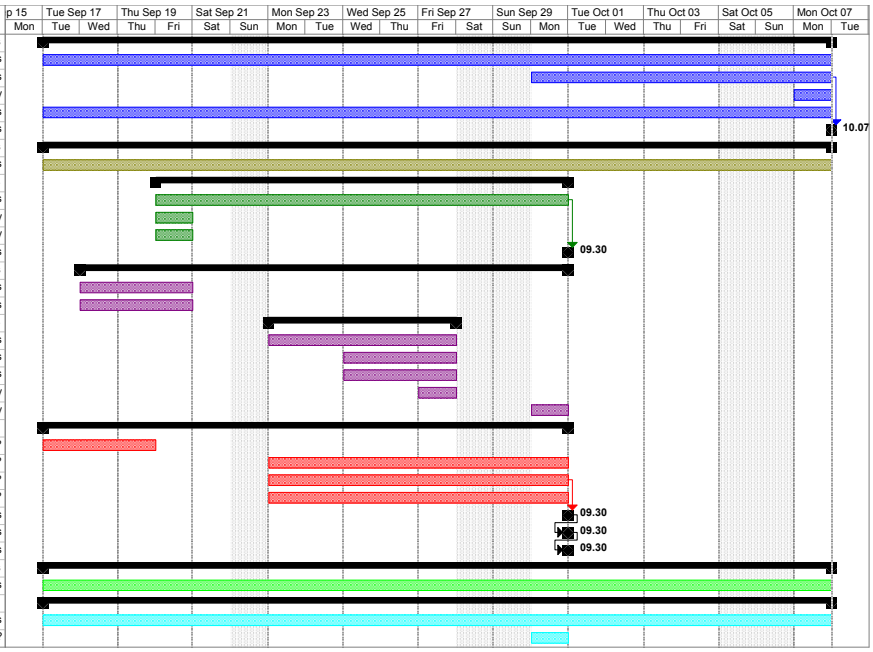
1.5 Plan for Iteration 3

1.5.1 Duration

17th of September – 7th of October

1.5.2 Overview

ID	Task Name	Duration
1	**Project Management**	**16 days**
2	Monitor & Control Project	16 days
3	Plan Next Iteration	6 days
4	Assess Iteration	1 day
5	Re-evaluate Scope & Risks	16 days
6	M14 - Iteration 4 Plan	0 days
7	**Requirements**	**16 days**
8	Manage Changing Requirements	16 days
9	**Analysis & Design**	**8 days**
10	Develop Software Architecture Document	8 days
11	Define Implementation Model	1 day
12	Define Logical View	1 day
13	M15 - SAD	0 days
14	**Implementation**	**10 days**
15	Implementation doc for Query System	3 days
16	Refactoring of Query System	3 days
17	**Implement Units**	**5 days**
18	GUI	5 days
19	Send/Receive Msg.	3 days
20	Send To A Specified Mobile User	3 days
21	Send Action Result	1 day
22	Integrate Code	1 day
23	**Test**	**11 days?**
24	Testing doc for Query System	3 days?
25	Prepare Test Units	6 days?
26	Implement Test Units	6 days?
27	Test Code	6 days?
28	M16 - Test Units	0 days
29	M17 - Messaging System	0 days
30	M18 - Query/Messaging System	0 days
31	**Environment**	**16 days**
32	Support Environment	16 days
33	**Configuration & Change Management**	**16 days?**
34	Change and update Documents	16 days
35	Manage Releases	1 day?



1.5.3 Deliverable Documents

- Test Units
- Iteration 4 Plan
- Software Architecture Document For Messaging System

1.5.4 Deliverable Releases

The Messaging System where the mobile user is able to send/receive messages to/from the central operator. All messages from the current session time is kept in memory and accessible to the user. The central operator can choose to which mobile user he/she wants to send messages to or can broadcast a message to all mobile users. All communications between all the mobile users and the central operator are displayed on the screen of the central operator. In addition to the normal messaging system then the mobile user is able to send Action Results related to a specific query result to the central operator, and this is then displayed on the screen of the central operator.

1.5.5 Milestones

Milestone: M14 – Iteration 4 Plan	**Date: 07/09/2002**
Description: At this stage, a detailed plan for the next iteration should have been created.	
Deliverables: Iteration Plans Document	
Evaluation Criteria: The Iteration Plans Document should be updated and should include a detailed plan for iteration 4.	

Milestone: M15 - Software Architecture Document	Date: 30/09/2002
Description: The Software Architecture Document for the Messaging System should be finalised	
Deliverables: SAD – Software Architecture Document	
Evaluation Criteria: The Software Architecture Document should include use case, data model, logical and implementation view of the Messaging System.	

Milestone: M16 – Test Units	Date: 30/09/2002
Description: All test units for the Messaging System should be ready at this stage.	
Deliverables: Test Units	
Evaluation Criteria: Test classes for every unit of the Messaging System, that can be tested with JUnit™ are ready.	

Milestone: M17 – Messaging System	Date: 30/09/2002
Description: The Messaging System with user interfaces, both for the central operator and the mobile user, is ready	
Deliverables: The Messaging System with all source codes	
Evaluation Criteria: The system has to meet the criteria set in *Section 1.5.4* and approved by the Quality Committee.	

Milestone: M18 – Query/Messaging System	Date: 30/09/2002
Description: The Query/Messaging System with the user interfaces, both for the central operator and the mobile user is ready.	
Deliverables: The Query/Messaging System with all its source code.	
Evaluation Criteria: An integrated version of the Query and the Messaging system with a common main GUI.	

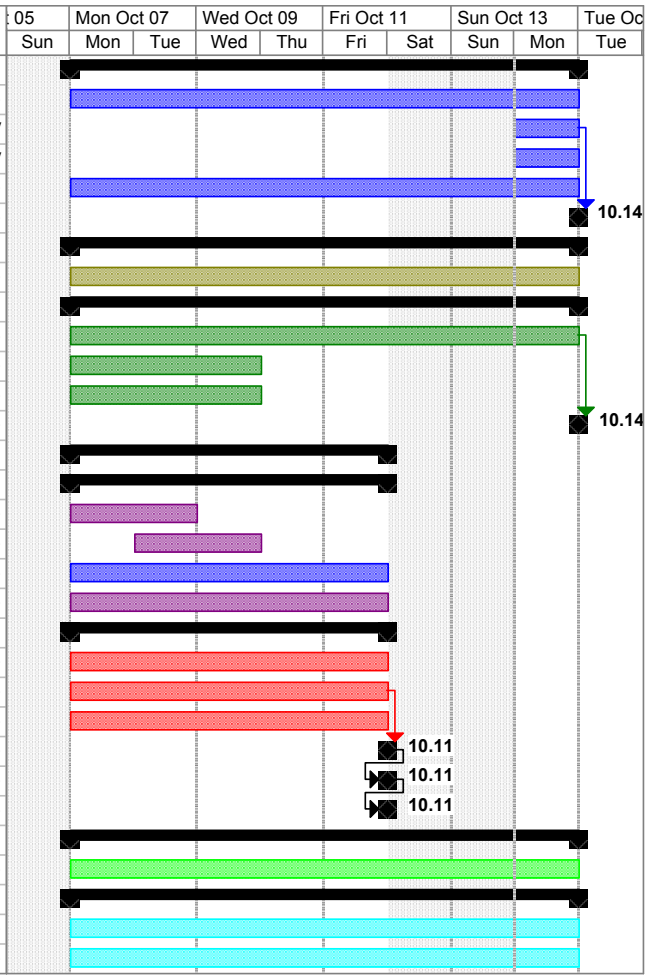
1.6 Plan for Iteration 4

1.6.1 Duration

7th of October – 14th of October

1.6.2 Overview

ID	Task Name	Duration
1	**Project Management**	**6 days**
2	Monitor & Control Project	6 days
3	Plan Next Iteration	1 day
4	Assess Iteration	1 day
5	Re-evaluate Scope & Risks	6 days
6	M23 - Iteration 5 Plan	0 days
7	**Requirements**	**6 days**
8	Manage Changing Requirements	6 days
9	**Analysis & Design**	**6 days**
10	Develop Software Architecture Document	6 days
11	Define Implementation Model	3 days
12	Define Logical View	3 days
13	M19 - SAD	0 days
14	**Implementation**	**5 days**
15	**Implement Units**	**5 days**
16	Mobile Client GIS display	2 days
17	Central Client GIS display	2 days
18	Refactoring	5 days
19	Integrate Code	5 days
20	**Test**	**5 days**
21	Prepare Test Units	5 days
22	Implement Test Units	5 days
23	Test Code	5 days
24	M20 - Test Units	0 days
25	M21 - Tracking System	0 days
26	M22 - The Car Tracking System	0 days
27	**Environment**	**6 days**
28	Support Environment	6 days
29	**Configuration & Change Management**	**6 days**
30	Change and update Documents	6 days
31	Manage Releases	6 days



1.6.3 Deliverable Documents

- Test Units
- Iteration 5 Plan
- Software Architecture Document For the Tracking System

1.6.4 Deliverable Releases

The Mobile User can see his/her own location on the AKIS[1] system. The Central Operator can see the location of all Mobile Users on the AKIS[1] system on his/her screen. However, because of absent of the GPS device we will work only with computer generated data regarding the locations of the vehicles.

[1] Akis®, V. Paliulionis

1.6.5 Milestones

Milestone: M19 - Software Architecture Document	Date: 14/10/2002
Description: The Software Architecture Document for the Tracking System should be finalised	
Deliverables: SAD – Software Architecture Document	
Evaluation Criteria: The Software Architecture Document should include use case, data model, logical and implementation view of the Tracking System.	

Milestone: M20 – Test Units	Date: 11/10/2002
Description: All test units for the Tracking System should be ready at this stage.	
Deliverables: Test Units	
Evaluation Criteria: Test classes for every unit of the Tracking System, that can be tested with JUnit™ are ready.	

Milestone: M21 – Tracking System	Date: 11/10/2002
Description: The Tracking System is ready	
Deliverables: The Tracking System with all source codes	
Evaluation Criteria: The system has to meet the criteria set in section 1.6.4 and approved by the Quality Committee.	

Milestone: M22 – The Car Tracking System	Date: 11/10/2002
Description: The Car Tracking System with the user interfaces, both for the central operator and the mobile user is ready, and communicates as needed with the AKIS[1] system.	

[1] Akis®, V. Paliulionis

Deliverables:
The Car Tracking System with all its source code.
Evaluation Criteria:
An integrated version of the Car Tracking System with a main GUI and running AKIS[1] parallel to it in order to display the positions of the cars.

Milestone: M23 – Iteration 5 Plan	**Date: 14/10/2002**
Description:	
At this stage, a detailed plan for the next iteration should have been created.	
Deliverables:	
Iteration Plans Document	
Evaluation Criteria:	
The Iteration Plans Document should be updated and should include a detailed plan for iteration 5.	

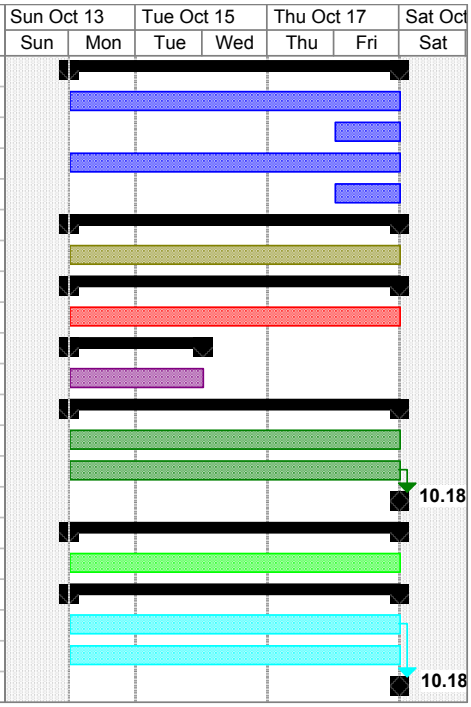
1.7 Plan for Iteration 5

1.7.1 Duration

14th of October – 18th of October

1.7.2 Overview

ID	Task Name	Duration
1	**Project Management**	**5 days**
2	Monitor & Control Project	5 days
3	Assess Iteration	1 day
4	Re-evaluate Scope & Risks	5 days
5	Close-Out Project	1 day
6	**Requirements**	**5 days**
7	Manage Changing Requirements	5 days
8	**Analysis and Design**	**5 days**
9	Refine Architecture	5 days
10	**Implementation**	**2 days**
11	Database Drivers Configurable	2 days
12	**Deployment**	**5 days**
13	Plan Deployment	5 days
14	Develop Deployment Document	5 days
15	M24 - Deployment Document	0 days
16	**Environment**	**5 days**
17	Support Environment	5 days
18	**Configuration & Change Management**	**5 days**
19	Change and update Documents	5 days
20	Manage Releases	5 days
21	M25 - Product Related Documents	0 days



[1] Akis®, V. Paliulionis

1.7.3 *Deliverable Documents*

- Deployment Document
- All prior product related documents mentioned in prior Iteration Plans

1.7.4 *Deliverable Releases*

A system with all the functionalities mentioned in prior Iteration Plans.

1.7.5 *Milestones*

Milestone: M24 – Deployment Document	**Date: 18/10/2002**
Description: The Deployment Document for the Car Tracking System should be finalised	
Deliverables: Deployment Document	
Evaluation Criteria: The Deployment Document should include a list of documents and files that are part of the deployment, an installation guide, guide for launching the applications for the Car Tracking System. It should also list all known errors and problematic features of the system.	

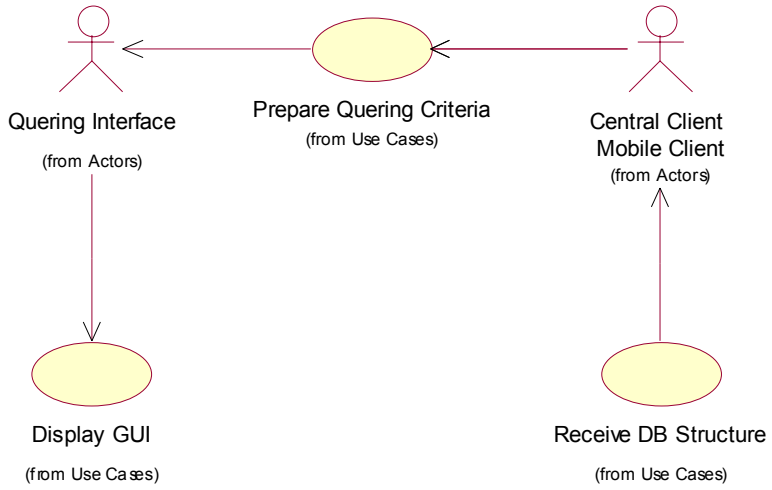
Milestone: M25 – Product Related Documents	**Date: 18/10/2002**
Description: All product related documents for the Car Tracking System should be finalised and up-to-date	
Deliverables: • Business Case • Vision Document • Software Requirements Specification • General System Architecture • Software Architecture Document; a general one and the for each subsystem • Use Case Specifications for each subsystem • Implementation Document for each subsystem • Test Document for each subsystem • Deployment Document	
Evaluation Criteria: All these documents should be finalised and up-to-date ready to be deployed to Sidabrinis Tinklas.	

Milestone: M26 – The Car Tracking System	Date: 18/10/2002
Description: The Car Tracking System demo is ready along with Java™ documentations of it.	
Deliverables: The source code of the Car Tracking System including its Java™ comments.	
Evaluation Criteria: A final version of the Car Tracking System from us, with Java™ comments for all the classes.	

Appendix C

1. Query System: Use Case Specifications

1.1 Generate GUI

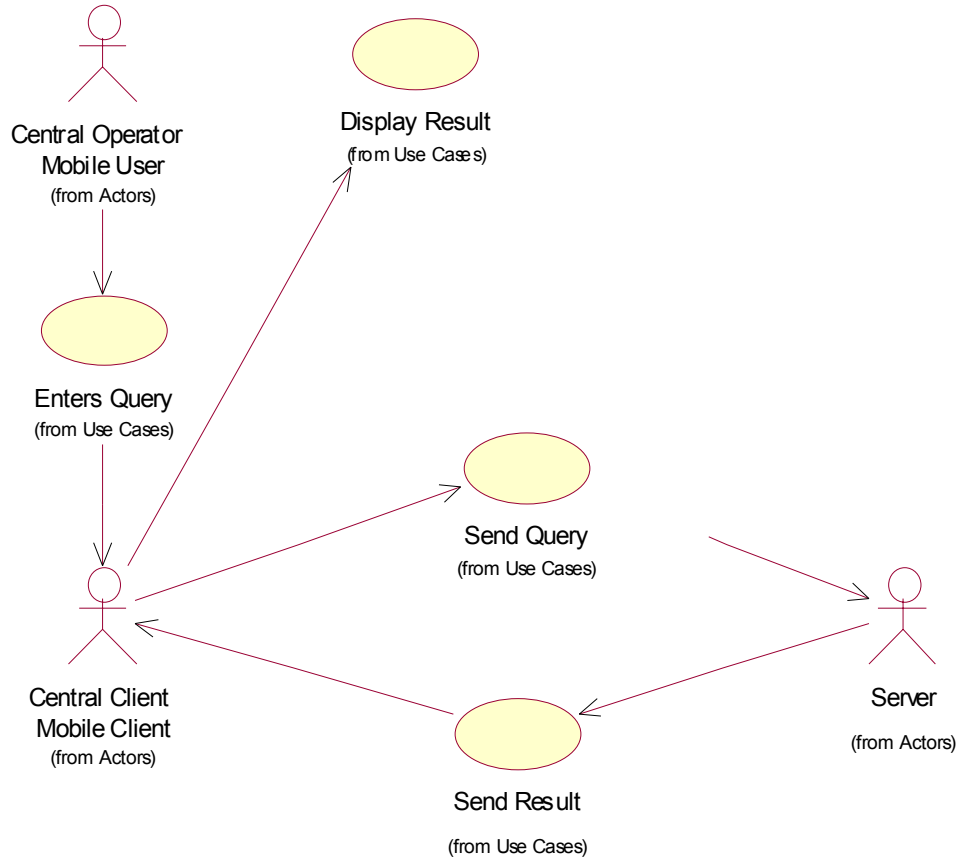


1.1.1 Details

Description
The Query Interface generates a GUI from the information given by the Central/Mobile Client. The information is created from the database structure and passed to the Central/Mobile Client.
Basic Flow
• The user selects a database to search in • The Central/Mobile Client receives the DB Structure • The Central/Mobile Client fills-in the info for searching GUI criteria • Query Interface reads info • Query Interface creates the GUI • The Use-Case ends
Alternative Flows
2. Structure Invalid *The DB structure passed to the Central/Mobile Client in not a valid structure. A message displaying the fatal error is displayed. The Query System closes.*
Pre – Conditions
None
Post – Conditions
A Graphical User Interface is created suiting the DB structure
Special Requirements
None

Frequency
Every time the query window is opened
Primary Actor
Querying Interface
Secondary Actor
Central/Mobile Client
Secondary Use-Cases
None

1.2 User Searches



1.2.1 Details

Description
The user needs information from the database. He/She enters in the search criteria and presses "Search". The query is formulated and sent to the server. The result(s) will then be sent back and displayed on the users screen.

Basic Flow

- The user enters in the search criteria
- The user presses “Search”
- The query is passed to the Central/Mobile Client
- The Central/Mobile Client sends the query to the Server
- The Central/Mobile Client receives the result(s) from the Server
- The result(s) is/are displayed
- The user presses close
- The Use-Case ends

Alternative Flows

1. Invalid criteria
   An error message appears telling the user that an invalid search value was entered. The user then has the possibility to re-enter the search criteria.

2. No matches
   A message appears telling that no matches were found; the user can then re-formulate the query.

3. Too many matches
   A message appears telling the user that too many matches are for the entered search criteria. The user gets the chance to narrow hi/her search.

4. *The Server is down*
   A message appears telling the user the problem and asking him/her t try again later.

Pre – Conditions

The user is logged on the system and has chosen the query system from the main menu.

Post – Conditions

The result(s) of the query is/are displayed

Special Requirements

None

Frequency

More than 10 times a day

Primary Actor

Central and Mobile Client

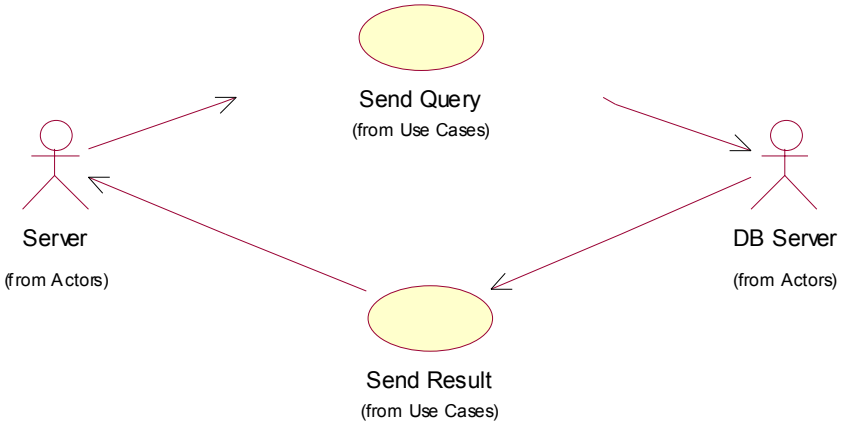
Secondary Actor

Server

Secondary Use-Cases

Server Queries The Database Server

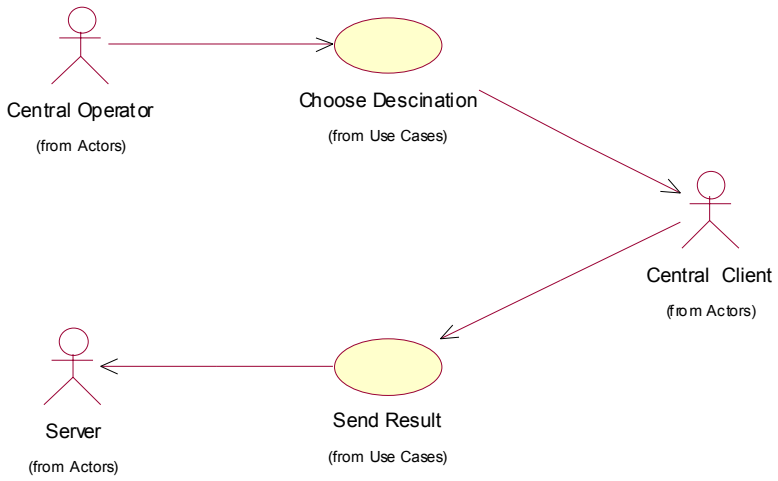
1.3 Server Queries The Database Server



1.3.1 Details

Description
The server receives a query from either the Central or the Mobile Client and forwards the query to and receives results from the Database server
Basic Flow
• The Central/Mobile Client send a query to the server, see *Use Case User Searches 1.2* • The Server forwards the query to the Database Server • The Database Server sends back the result(s) from the query • The Server receives the result(s) and forwards it to the Central/Mobile Client, see *Use Case User Searches 1.2* • The use-case ends.
Alternative Flows
1. The Database Server is down *The main server gets a error message and forwards it to the Central/Mobile Client*
Pre – Conditions
Query has been sent from the Central/Mobil Client
Post – Conditions
The main server gets all the results asked for
Special Requirements
None
Frequency
More than 50 time a day
Primary Actor
Server
Secondary Actor
Database server, Central/Mobile client
Secondary Use-Cases
None

1.4 Central Operator Sends Query With No Request



1.4.1 Details

Description
The Central Operator wishes to send a query result that is already displayed on the screen of the Central Operator to a specific client
Basic Flow
• Central Operator chooses a client destination • Central Client sends the message to the server • The use-case ends.
Alternative Flows
1. Communication Drops An error message is displayed and shows the connection problem.
Pre – Conditions
• The Central Operator is logged on • The Central Operator has the required Query result already displayed
Post – Conditions
The Query Result is displayed at the Mobile Users screen
Special Requirements
None
Frequency
Less than 50 times a day
Primary Actor
Central Operator
Secondary Actor
Central Client
Secondary Use-Cases
User Searches, Server Queries The Database Server.

Appendix D

1. Query System: Software Architecture Document

1.1 Server Component

1.1.1 Function Specification

Name	Server.validateDBStructure()
Input	Database Object
Output	Boolean

Algorithm	Do{
	Read **line** from file;
	If (**line** is not empty){
	Read **token**;
	In case **token** is:
	"<TABLE>"{
	ok;
	}
	"TABLENAME"{
	if (TABLE tag is open){
	read next **token**;
	if (**token** exist) **ok**
	else **error**;
	}
	}
	"</TABLE>"{
	if (No fields declared) **error;**
	if (No tablename declared) **error;**
	if (Table tag is open) **ok;**
	}
	"<FIELD>"{
	if (Table tag has been opened) **ok;**
	else **error;**
	}
	"</FIELD>"{
	if (Field tag has not been opened) **error;**
	if (Name, Type and Searchable don't exist)
	error;
	else **ok;**
	}
	"NAME"{
	if (Field tag has been opened){
	read next **token;**
	if (**token** is empty) **error;**
	else **ok:**
	}
	}
	"TYPE"{

	if (Field tag has been opened){
	read next **token;**
	if (**token** is empty) **error;**
	else {
	if (**token** valid) **ok**
	else **ok**
	}
	}
	}
	"SEARCHEABLE"{
	if (Field tag has been opened){
	read next **token;**
	if (**token** is empty) **error;**
	else **ok:**
	}
	
	}
	}
	
	}
	while (no more line on text file)
	if (only one table has been defined AND the table tag is closed) **ok;**
	else **error;**

Name	Server.translateQuery()
Input	Query Object
Output	SQL Query

Algorithm	`String query = "SELECT * FROM "`
	`Append tablename to query`
	`Append " WHERE " to query`
	
	`For (each search criteria inserted)`
	`Append field_name to query`
	`Append search criteria("=" or "LIKE")`
	`Append field_value to query`
	`End For loop`
	
	`Return query`

1.2 Mobile Client Component

1.2.1 Function Specification

Name	QueryGUI.createQuery()
Input	Search criteria from the GUI
Output	Query Object

Specification	Save every searched field with search criteria and value in the Query object. When the Query Object is created, the database, where the search has been made, must be passed to the Query Object.

1.3 Central Client Component

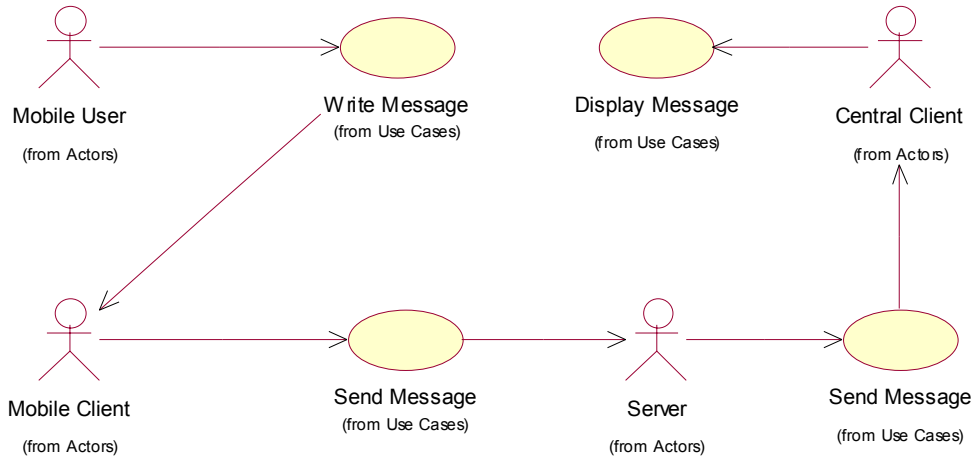
1.3.1 Function Specification

Name	ResultGUI.sendResult()
Input	Mobile Client
Output	Result
Specification	The Result and the Mobile Client data are sent to the Server, which will then dispatch the Result to the Mobile Client Application.

Appendix E

1. Messaging System: Use Case Specifications

1.1 Mobile User Message Sending

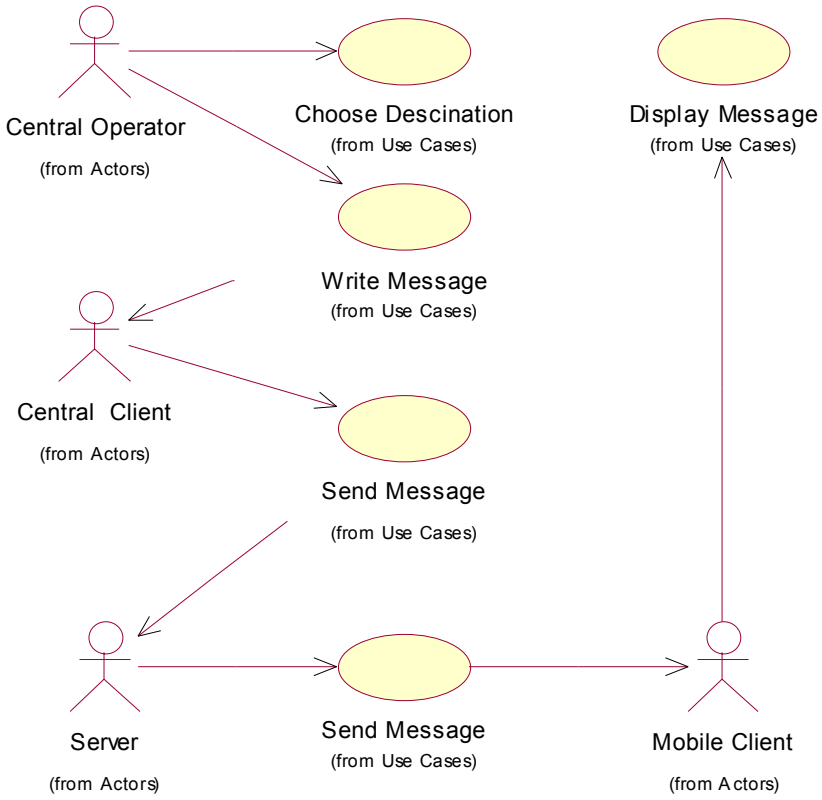


1.1.1 Details

Description
The Mobile User has to send a message to the Central Client. He/she writes the message and sends it through the Mobile Client. The Server receives the message and dispatches it to the Central Client.
Basic Flow
• The Mobile User selects the messaging text area • The Mobile User writes a message • The Mobile Client sends the message to the Server • The Server receives the message • The Server sends the message to the Central Client • The Central Client receives the message • The Central Client displays the message • The use-case ends
Alternative Flows
3. *Server is Down* An error message is displayed, asking to try again later. 4. *Central Client in not on-line* An error message is displayed at the Mobile Client side, acknowledging the failure of sending the message
Pre – Conditions
The Mobile User is logged on

Post – Conditions
A message is received and displayed at the Central Client
Special Requirements
None
Frequency
More than 10 time a day
Primary Actor
Mobile User
Secondary Actor
Server, Central Client
Secondary Use-Cases
None

1.2 Central Operator Message sending

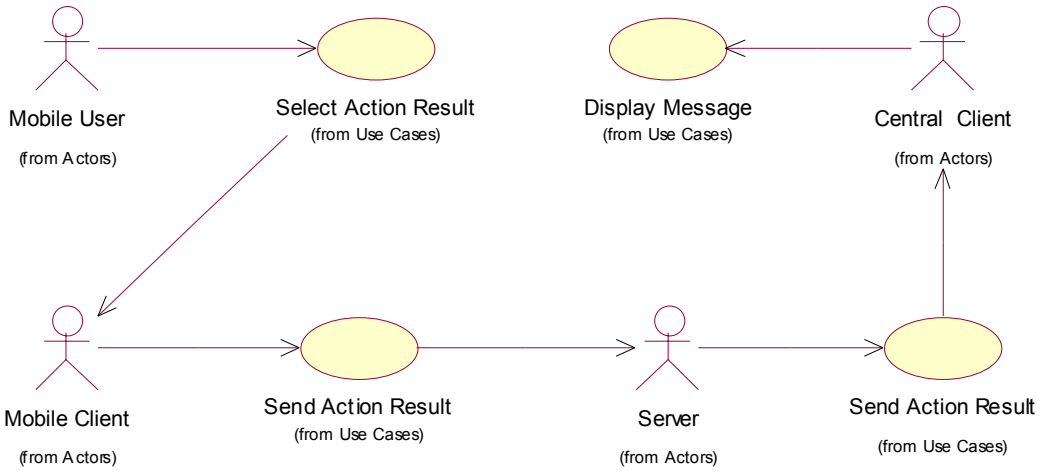


1.2.1 Details

Description
The Central Operator writes a message and sends it to a specific client or broadcast it. The Mobile Client receives the message and displays it.

Basic Flow
• Central Operator chooses the destination (client / broadcast) • The text area activates • Central Operator writes the message • Central Client sends the message to the Server • Server sends the message to the specific Mobile Client or to all • Mobile Client(s) receive(s) the message • Mobile Client display the message • The use-case ends
Alternative Flows
1. *Communication Drops* An error message is displayed and shows the connection problem 2. *Server is down* An error message is displayed at the Central Client side, acknowledging the failure of sending the message
Pre – Conditions
The Central Operator is logged on
Post – Conditions
A message is displayed at the Mobile Client
Special Requirements
None
Frequency
More than 20 times a day
Primary Actor
Central Operator
Secondary Actor
Server, Mobile Client
Secondary Use-Cases
None

1.3 Mobile User Action Result Sending



1.3.1 Details

Description
The Mobile User has to send an Action Result to the Central Client. The Mobile User selects the Action Result and sends it through the Mobile Client. The Server receives the message and sends it to the Central Client
Basic Flow
• The Mobile User selects the Action Result • The Mobile Client sends the Action Result to the Server • The Server Receive the Action Result • The Server sends the Action Result to the Central Client • The Central Client receives the Action Result • The Central Client displays the Action Result • The use-case ends
Alternative Flows
1. *Server is Down* An error message is displayed, asking to try again later. 2. *Central Client in not on-line* An error message is displayed at the Mobile Client side, acknowledging the failure of sending the message
Pre – Conditions
The Mobile User is Logged on
Post – Conditions
A message is received and displayed at the Central Client
Special Requirements
None

Frequency
More than 10 time a day
Primary Actor
Mobile User
Secondary Actor
Server, Central Client
Secondary Use-Cases
None

Appendix F

1. Messaging System: Software Architecture Document

1.1 Server Component

1.1.1 Function Specification

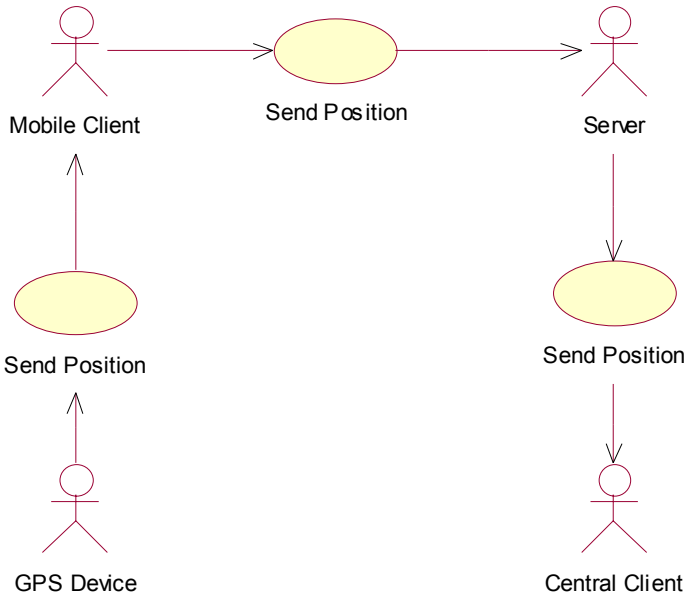
Name	Server.save_Action()
Input	The Action Object
Output	Updated Database
Algorithm	The function is not inside the scope of the demo version therefore we will not present the algorithm for it

Name	Server.save_Message()
Input	The Message Object
Output	Updated Database
Algorithm	The function is not inside the scope of the demo version therefore we will not present the algorithm for it

Appendix G

1. Tracking System: Use Case Specifications

1.1 Mobile Client Sends Position To Central Client

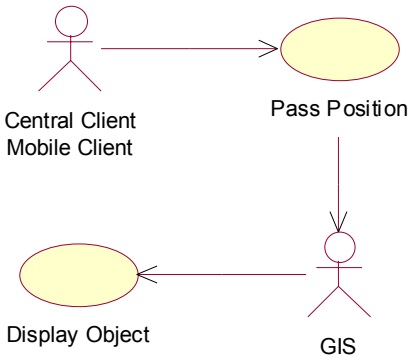


1.1.1 Details

Description
The Mobile Client receives the position from the GPS device and sends the position to the Central Client through the Server
Basic Flow
• The GPS Device sends the position to the Mobile Client • The Mobile Client receives the position and send it to the Server • The Server receives the position • The Server sends the position to the Central Client • The Central Client receives the position • End of the use-case
Alternative Flows
1. Server is Down *An error message is displayed, asking to try again later.*

Pre – Conditions
None
Post – Conditions
The Mobile Client position is received by the Central Client
Special Requirements
None
Frequency
Every 2 seconds
Primary Actor
Mobile Client
Secondary Actor
Server, GPS Device, Central Client
Secondary Use-Cases
Display Position

1.2 Display Position

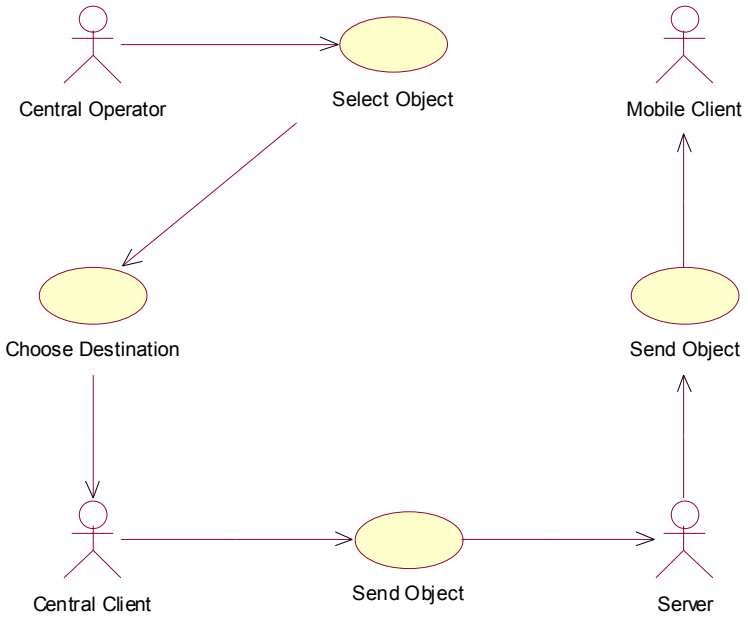


1.2.1 Details

Description
The Mobile/Central Client passes the Position of the Object to the GIS, which will then display the Object
Basic Flow
• The Mobile/Central Client passes the Object's position to the GIS • The GIS receives the Object's position • The GIS displays the Object • End of the use-case
Alternative Flows
None
Pre – Conditions
The Mobile/Central Client has received an Object's position
Post – Conditions
The Object is displayed

Special Requirements
None
Frequency
Every 2 seconds
Primary Actor
Mobile Client/Central Client
Secondary Actor
GIS
Secondary Use-Cases
None

1.3 Central Client Sends An Object



1.3.1 Details

Description
The Central Operator selects an Object to be sent to a Mobile User; the Object is then sent and displayed at the Mobile Client
Basic Flow
• The Central Operator selects the Object to send • The Central Operator chooses the Destination • The Central Client sends the Object to the Server • The Server dispatches the Object to the destination Mobile Client • The Mobile Client receives the Object • End of the use-case

Alternative Flows
1. Server is Down *An error message is displayed, asking to try again later.*
Pre – Conditions
None
Post – Conditions
The Object is displayed using the GIS on the Mobile Client side
Special Requirements
None
Frequency
More than 5 times a day
Primary Actor
Central Operator
Secondary Actor
Server, Mobile Client, Central Client
Secondary Use-Cases
Display Position

Appendix H

1. Messaging System: Software Architecture Document

1.1 Mobile Client Component

1.1.1 Function Specification

Name	GPSDevice.read_Position()
Input	None
Output	The Position read by from the GPS Device
Algorithm	Specification of this function is left for future development, as the actual device was not available at this stage.

Name	GIS.show_Map()
Input	None
Output	Updated Map
Algorithm	The integration of the GIS system into the Car Tracking System is left for future development; therefore, the specification of this function has not been made.

Name	GIS.updateObject() ; GIS.createObject(); GIS.deleteObejct();
Input	A GIS Object
Output	Updated GIS database
Algorithm	All of these functions are used to insert, update and delete records from the database file “Objektai.dbf” of the GIS. This file is used as communication mean with the GIS software. The Objektai table is designed as following: NAME | LON | LAT The field used as primary key is the “NAME”, while “LON” represents the longitudinal coordinate, and “LAT” the latitudinal coordinate of the Object.

Appendix I

1. Deployment Document

1.1 Introduction

1.1.1 Purpose

The purpose of the Deployment Document is to provide an overview of what comes with the hand over of the project, both software and documents.

1.1.2 Scope

The scope is the Car Tracking System Project as a whole. It is should be of value for future developers, and users of the demo version.

1.1.3 Overview

First, we will list out what is included in the deployment unit, which is handed over to Sidabrinis Tinklas that includes both the software and the documentations. Then we will cover the installation and configuration instructions, explaining everything that is needed to run the demo version itself. Finally we discuss the errors and other problematic features of the demo, know at release time.

1.2 The Deployment Unit Description

1.2.1 Inventory of Materials

Documents

No.	Name	Version	Format
1.	Business Case	1.1	Word Document (*.doc)
2.	Demo Development Proposal	1.1	Word Document (*.doc)
3.	Deployment Document	1.3	Word Document (*.doc)
4.	Development Case	3.0	Word Document (*.doc)
5.	General System Architecture	1.0	Word Document (*.doc)
6.	Implementation Document	1.0	Word Document (*.doc)
7.	Iteration Assessments	4.0	Word Document (*.doc)
8.	Iteration Plans	4.0	Word Document (*.doc)
9.	Query System Design Proposals	2.0	Word Document (*.doc)
10.	Risk List	1.4	Word Document (*.doc)
11.	Software Architecture Document, General	2.0	Word Document (*.doc)
12.	Software Architecture Document, Messaging System	1.1	Word Document (*.doc)
13.	Software Architecture Document, Query System	2.2	Word Document (*.doc)
14.	Software Architecture Document, Tracking System	1.2	Word Document (*.doc)
15.	Software Development Plan	3.0	Word Document (*.doc)

No.	Name	Version	Format
16.	Software Requirement Specification	1.8	Word Document (*.doc)
17.	Test Document For Messaging System	1.0	Word Document (*.doc)
18.	Test Document For Query System	1.0	Word Document (*.doc)
19.	Test Document For Tracking System	1.0	Word Document (*.doc)
20.	Use Case Specifications	3.4	Word Document (*.doc)
21.	Vision Document	3.2	Word Document (*.doc)

Models

No.	Name	Format
1.	Analysis_model_Message	Rational Rose® (*.MDL)
2.	Analysis_model_Query	Rational Rose® (*.MDL)
3.	Analysis_Model_Tracking_Sys	Rational Rose® (*.MDL)
4.	Architecture	Rational Rose® (*.MDL)
5.	Design_model_query	Rational Rose® (*.MDL)
6.	Design_model_tracking_sys	Rational Rose® (*.MDL)
7.	General Architecture	Rational Rose® (*.MDL)
8.	Implementation_Model_Message	Rational Rose® (*.MDL)
9.	Implementation_Model_Query	Rational Rose® (*.MDL)
10.	Implementation_model_tracking_sys	Rational Rose® (*.MDL)
11.	Tracking_System_Use_cases	Rational Rose® (*.MDL)

Software

No.	Name	Format
1.	centralclient	Executable File (*.exe)
2.	Installation	Text file (*.txt)
3.	mobileclient	Executable File (*.exe)
4.	server	Executable File (*.exe)
5.	Source	Zip file (*.zip)
6.	Test	Zip file (*.zip)

1.2.2 Inventory of Software Contents

Server.exe

No.	Name	Format
1.	config	Data file (*.dat)

No.	Name	Format
2.	database	Data file (*.dat)
3.	DBStructure	Text file (*.txt)
4.	mysql-connector-2.0.14-bin	Java™ Archive File (*.jar)
5.	ServConf	Initialisation File (*.ini)
6.	Server	Java™ Archive File (*.jar)

Remarks

- mysql-connector-2.0.14-bin.jar is a JDBC[1] driver used only for testing
- Server.jar contains the class files needed to run the server but the list of all the Java™ files from which these files were made can be seen in 2.2.5

Centralclient.exe

No.	Name	Format
1.	Back24	Graphic Interchange Format (*.gif)
2.	CentralClient	Java™ Archive File (*.jar)
3.	Forward24	Graphic Interchange Format (*.gif)

Remarks

- The centralclient.exe also contains needed files to run the Akis[2] GIS Demo
- CentralClient.jar contains the class files needed to run the server but the list of all the Java™ files from which these files were made can be seen on *page XLIII.*

Mobileclient.exe

No.	Name	Format
1.	Back24	Graphic Interchange Format (*.gif)
2.	Forward24	Graphic Interchange Format (*.gif)

[1] Sun Microsystems

[2] Akis© V.Paliulionis

No.	Name	Format
3.	MobileClient	Java™ Archive File (*.jar)

Remarks

- The mobileclient.exe also contains needed files to run the Akis[1] GIS Demo
- CentralClient.jar contains the class files needed to run the server but the list of all the Java™ files from which these files were made can be seen on *page XLIII*.

Installation.txt

No.	Name	Format
1.	Installation	Text file (*.txt)

Source.zip

No.	Name	Format
1.	Action	Java file (*.Java)
2.	CentralClient	Java file (*.Java)
3.	Client	Java file (*.Java)
4.	ClientServerCommunication	Java file (*.Java)
5.	CMainGUI	Java file (*.Java)
6.	CMessageGUI	Java file (*.Java)
7.	Config	Java file (*.Java)
8.	Coordinate	Java file (*.Java)
9.	CResultGUI	Java file (*.Java)
10.	CsearchPanel	Java file (*.Java)
11.	CTS_Server	Java file (*.Java)
12.	Database	Java file (*.Java)
13.	DBServerCommunication	Java file (*.Java)
14.	DBStructure	Java file (*.Java)
15.	DBStructure1	Java file (*.Java)
16.	EBConnection	Java file (*.Java)
17.	EBCSCommunication	Java file (*.Java)
18.	EBSCCommunication	Java file (*.Java)
19.	Field	Java file (*.Java)
20.	GIS	Java file (*.Java)

No.	Name	Format
21.	GISCommunication	Java file (*.Java)
22.	GISObject	Java file (*.Java)
23.	GPSCommunication	Java file (*.Java)
24.	GPSDevice	Java file (*.Java)
25.	Info	Java file (*.Java)
26.	InvalidStructureException	Java file (*.Java)
27.	MainGUI	Java file (*.Java)
28.	Message	Java file (*.Java)
29.	MessageGUI	Java file (*.Java)
30.	MobileClient	Java file (*.Java)
31.	MobileClientThread	Java file (*.Java)
32.	NotEnoughParameterException	Java file (*.Java)
33.	OnLineUsersGUI	Java file (*.Java)
34.	Query	Java file (*.Java)
35.	QueryField	Java file (*.Java)
36.	QueryGUI	Java file (*.Java)
37.	Result	Java file (*.Java)
38.	ResultCache	Java file (*.Java)
39.	ResultField	Java file (*.Java)
40.	ResultGUI	Java file (*.Java)

[1] Akis© V.Paliulionis

No.	Name	Format
41.	ResultInfo	Java file (*.Java)
42.	ScrollablePicture	Java file (*.Java)
43.	SDatabase	Java file (*.Java)
44.	SearchPanel	Java file (*.Java)
45.	Server	Java file (*.Java)
46.	ServerClientCommunication	Java file (*.Java)

No.	Name	Format
47.	ServerThread	Java file (*.Java)
48.	SImage	Java file (*.Java)
49.	SysConst	Java file (*.Java)
50.	Table	Java file (*.Java)
51.	User	Java file (*.Java)
52.	Users	Java file (*.Java)

Test.zip

No.	Name	Format
1.	CTSAllTests	Java file(*.Java)
2.	DatabaseTest	Java file (*.Java)
3.	EBConnectionTest	Java file (*.Java)
4.	FieldTest	Java file (*.Java)
5.	MessageTest	Java file (*.Java)
6.	ResultTest	Java file (*.Java)
7.	ServerTest	Java file (*.Java)
8.	TableTest	Java file (*.Java)
9.	DBStructure	Text file (*.txt)
10.	DBStructure1	Text file (*.txt)
11.	DBStructure_test1	Text file (*.txt)
12.	DBStructure_test2	Text file (*.txt)
13.	DBStructure_test3	Text file (*.txt)
14.	DBStructure_test4	Text file (*.txt)
15.	DBStructure_test5	Text file (*.txt)
16.	DBStructure_test6	Text file (*.txt)
17.	DBStructure_test7	Text file (*.txt)
18.	DBStructure_test8	Text file (*.txt)
19.	DBStructure_test9	Text file (*.txt)
20.	DBStructure_test10	Text file (*.txt)
21.	DBStructure_test11	Text file (*.txt)
22.	DBStructure_test12	Text file (*.txt)
23.	DBStructure_test13	Text file (*.txt)
24.	Read.me	Text file (*.txt)

1.3 Installation And Configuration Instructions

1.3.1 System Requirement

- Java™ SDK 1.4.0 must be installed
- The CLASSPATH must be a be to reach the installation directories

1.3.2 Installation

Server

- Choose or create the directory that will hold the Server
- Extract the Server.zip file in the selected directory
- Configure the server using the ServConf.ini file - see Server configuration at 1.3.3

Central Client

- Choose or create the directory that will hold the Central Client
- Extract the CentralClient.zip file in the selected directory
- Create a Data Source for the GIS System under “System DSN”
    - Data Source Name must be “GIS_Object “
    - Data Source Driver must be “Microsoft Driver for Visual FoxPro”
    - Data Source Database type must be set to “Free table directory”
    - Working Directory must be the [the_Central_Client_Directory]/Akis GIS Demo/VILNIUS

Mobile Client

- Choose or create the directory that will hold the Mobile Client
- Extract the MobileClient.zip file in the selected directory
- Create a Data Source for the GIS System under “System DSN”
    - Data Source Name must be “GIS_Object”
    - Data Source Driver must be “Microsoft Driver for Visual FoxPro”
    - Data Source Database type must be set to “Free table directory”
    - Working Directory must be the [the_Mobile_Client_Directory]/Akis GIS Demo/VILNIUS

1.3.3 Run And Configure The System

The Server

All configuration parameter of the Server are in the ServConf.ini file, to configure the Server it is only needed to change the value of the parameters and restart the server with the '-lc' option

- Run the server:

    Java™ -jar Server.jar

- Run the server with a new configuration:

    Java™ -jar Server.jar -lc

The ServConf.ini

- filename

  The name and path of the file containing the database structure e.g. “DBStructure.txt”

- dbDriver

  The driver class for JDBC[1] e.g. “sun.jdbc.odbc.JdbcOdbcDriver”

- dbURL

  The URL or path of the database e.g. “jdbc:odbc:EKM”

- dbName

  The name of the database to be used e.g. “EKM”

- username

  username to access the database e.g. “dario64”

- password

  The password to access the database e.g. “darion65”

Note:

The username and password are the only fields that can be left without value.

The Mobile Client

- Run the mobile client:

  Java™ -jar MobileClient.jar *[User Name] [server IP address]*

  - *[User Name]*: Used to identify the client, make sure it is unique
  - *[server IP address]*: The IP address where the server is running for local connection 127.0.0.1 can be used

The Central Client

- Run the central client:

  Java™ -jar CentralClient.jar *[server IP address]*

  - *[server IP address]*: The IP address where the server is running for local connection 127.0.0.1 can be used

1.4 Known Errors and Problematic Features

1.4.1 NullPointerException

Occasionally, when the Mobile Client application is run, a “NullPointerException” is thrown at the Server side. The Exception though does not cause any system failure.

1.4.2 Exception Thrown From The Tracking System

When the Central Client is closed before the Mobile Clients an exception is thrown at the Mobile Client and at the Server side, but the system seems to work anyway.

[1] Sun Microsystems

1.4.3 SQL Error

In `EBConncetion` class, in the `translateQuery(…);` method, the query created makes a select query to a table, but as it is working with a DBMS it will have to specify the database on which to work on. To do this then we send a "USE *database_name*" SQL sentence. This will though not work on DBMS that does not accept this command (e.g. Microsoft Access®).

Appendix J

1. The Protocol

The follow info table describes the protocol used for the transmission of data between the clients and the server.

Command Token	Comment	Expected Data
1	Send a query request	One Query object
2	Receive a result object	Mix of ResultField objects and Images. Commands: 4: A new row of fields 5: An image stream is coming 3: The transmission end Note: Images are sent byte by byte, starting with the number of bytes that will be sent.
3	Start client shutdown procedure	None
4	Send Database Structure	One Database object
5	Notify that a client logged in	One User object
6	Send Client Information	One Info object
7	Notify that User logged out	One User object
8	Send a message	One Message object
10	Send the car position	One GISObject object

Appendix K

1. Effort Sheets

Name: Dario Pacino		**Project:** Car Tracking System					**Week ending:** 18/08/2002			
No	Task	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total	To Go
1	Finalise Business Case Document	0,5							0,5	Nil
2	Write Software development plan	3,0								
3	Plan Project Phases	1,0							1,0	Nil
4	Plan First Iteration	1,0							1,0	Nil
5	Study Architecture		1,0						1,0	Nil
6	Define Advantages and Disadvantages of Architecture structure		2,0						2,0	Nil
7	Choose Architecture		1,0						1,0	Nil
8	Make Use Case		1,0						1,0	Nil
9	Update Software Development Plan			1,0					1,0	Nil
10	Study System Possible Solutions			4,0					4,0	Nil
11	Meeting			0,5					0,5	Nil
12	Prepare Iteration 1 plan						1,0		1,0	1,0
13	Prepare Solutions Document						4,0		4,0	Nil

Name: Hjortur Scheving		**Project:** Car Tracking System					**Week ending:** 18/08/2002			
No	Task	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total	To Go
1	Review Documents			1.5						
2	Meeting			0.5					0.5	Nil
3	Evaluation Meeting Document			0.5					0.5	Nil
4	Creating Templates			1						
5	Editing Documents			4.5			5	5	14.5	?

Name: Dario Pacino		**Project:** Car Tracking System					**Week ending:** 25/08/2002			
No	Task	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total	To Go
1	Prepare Iteration 1 Plan	1.0							2.0	Nil
2	Editing	4.0							4.0	Nil
3	Meeting	0.5							0.5	Nil
4	Report Structure		1.0						1.0	Nil
5	Write System Architecture Chapter for Report		2.0						2.0	Nil
6	Write Risk List Task For Report		1.0						1.0	Nil

Appendices

Name: Hjortur Scheving	**Project:** Car Tracking System				**Week ending:** 25/08/2002					
No	Task	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total	To Go
1	Software Development Plan	8	1							
2	Meeting	1	0.1							
3	Organising hard copies		1							
4	Problem Formulation		2	1						
5	Report Structure		1							
6	Research Methodologies			2	2	3	4	2		
7	Write Methodologies							2		
8	Editing Report			2	2					
9	Writing For Report, Software D.P.				2					
10	Write For Report, Vision, Business Case			1						

Name: Dario Pacino	**Project:** Car Tracking System				**Week ending:** 01/09/2002					
No	Task	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total	To Go
1	Use-case Spec. for Query Sys	3							3	Nil
2	SRS for Query Sys	4							4	Nil
3	Use-Cases for Msg. Sys		3						3	Nil
4	SRS for Msg. Sys		3						3	Nil
5	Update Risk List		1						1	Nil
6	Fix Quality Range on Vision			1.0					1	Nil
7	Fix Layers text			0.5					0.5	Nil
8	Fix SRSs			1.5					1.5	Nil
9	Make Logical View			2.5					2.5	Nil
10	Make GUI Skaches			3.0	1.0				4.0	Nil
11	Define Architecture				1.0					3.0
12	Revise SRS for USER				3.0				3.0	Nil
13	Analysis & Design section 1					6.0		1.0	7.0	Nil

Name: Hjortur Scheving	**Project:** Car Tracking System				**Week ending:** 01/09/2002					
No	Task	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total	To Go
1	Putting RUP® Doc into the report		11	1	2				14	On going
2	Building the new report structure and layout	2			1	5	1		9	Nil
3	Meetings		1		0.5				1.5	
4	Problem Formulation, update	5							5	Nil
5	Logical View			1					1	
6	Report, adding SRS			2					2	
7	Report Editing			2	1	1			4	On going
8	Development Case			2	7				9	8
9	GUI Sketches			1					1	On Going
10	Iteration 0 Assessment						2		2	1

L

Name: Dario Pacino	**Project:** Car Tracking System				**Week ending:** 08/09/2002					
No	Task	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total	To Go
1	Modify SAD	2.5							2.5	Nil

Name: Hjortur Scheving	**Project:** Car Tracking System				**Week ending:** 08/09/2002					
No	Task	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total	To Go
1	New SRS – General	0.5								Nil
2	New SRS – Query System	1								Nil
3	New SRS – Messaging System	1								Nil
4	Iteration 0 Assessment	1								Nil

Appendix L

1. Evaluation Meeting 1

1.1 Introduction

1.1.1 Purpose

To get Quality Group's evaluation of documents presented (see Documents Reviewed). And discuss the next steps of the project.

1.1.2 Date and Time

14th of August 2002, 10.30-11.00

1.1.3 Venue

Sidabrinis Tinklas, meeting room

1.1.4 Participants

Giedrius Slivinskas
Dario Pacino
Hjortur Scheving

1.2 Actions

1.2.1 Documents Reviewed

- Vision *Draft*
- Business Case *<1.0>*
- Software Development Plan *<1.0>*
- System Architecture *<1.0>*
- Risk List *<1.0>*
- General Plan *Draft*
- Iteration 0 – Plan *Draft*

1.2.2 Documents to be Revised

- Software Development Plan *<1.0>*
  - Clarify deliverables
- Risk List *<1.0>*
  - Practical Risks – minor changes

1.2.3 Next Steps

- Iteration 1 – Plan
- Query System Options document
  - Covering the options we have with the Query system, with special focus on how to;
    - receive DB structure
    - implement interface from the DB structure
    - generate queries from the interface
    - displaying the results
  - The evaluation criteria for the options listed in the document should be;
    - How long will it take to add new DB to the system?
    - How long will it take to implement?
    - How well will it satisfy the customer's needs?
- Query System Proposal document

    - Covering the proposed solution from the Option document in a short, direct way. To be handed to the user company
- Interface Specification

2. Evaluation Meeting 2

2.1 Introduction

2.1.1 Purpose

To get Quality Group's evaluation of documents presented (see Documents Reviewed). And discuss the next steps of the project.

2.1.2 Date and Time

19th of August 2002, 14.00-15.00

2.1.3 Venue

Sidabrinis Tinklas, meeting room

2.1.4 Participants

Giedrius Slivinskas
Dario Pacino
Hjortur Scheving

2.2 Actions

2.2.1 Documents Reviewed

- Iteration Plan *<2.0>*
- Query System Solution Options (hand written copy)

2.2.2 Documents to be Revised

- Iteration Plan <2.0>
    - A more detailed plan
- Query System Solution Options
    - Fix time estimates
    - Add and modify ad- and disadvantages

2.2.3 Next Steps

- Query System Project Proposal document
    - Listing
        - The SQL solution
        - The XML solution
        - Discussing for both solutions:
            - Ad- and disadvantages
            - How long does it take to add a DB for the User
            - Constrains
            - Etc.
- Software Requirements Specification

3. Evaluation Meeting 3

3.1 Introduction

3.1.1 Purpose

To go over a meeting that Giedrius had with the user company and clarify a few uncertain requirements. And discuss the next steps of the project.

3.1.2 Date and Time

18th of August 2002, 14.00-15.00

3.1.3 Venue

Sidabrinis Tinklas, meeting room

3.1.4 Participants

Giedrius Slivinskas
Dario Pacino
Hjortur Scheving

3.2 Actions

3.2.1 Documents Reviewed

- *EKMRTPTO002_0100_meeting_notes_EB_20020827.doc*

3.2.2 Next Steps

- As listed in the EKMRTPTO002_0100_meeting_notes_EB_20020827.doc

3.3 Notes

The following was discussed:

- Action Results: actions related to objects
- HTTP Server: has to be implemented by the development team
- Authentication control: needs to be address during the prototype
- Keyboards: Should be given some thought, specially when designing the GUI as a keyboard screen may be displayed in the GUI

4. Meeting Between ST and EB

Meeting notes
2002 08 27 @ EB
Participants: Giedrius Slivinskas (ST), Alius Sadeckas (EB)

Software development phases and steps:

Phase I – Query system

1. Mobile users send queries to the central system and receive feedback.
2. Mobile users send action results to the central system, e.g., describe what happened (the result may be a selected option from the list or a field with user input)
3. Central system sends query answers to mobile users without request; the answer is displayed on the client application.
4. Central system sends messages to mobile users
5. Mobile users send messages to the central system

Phase II – GIS

1. Mobile users send their GPS positions to the central system. The central system displays these positions on a GIS system.
2. Mobile user sees his/her positions on a car GIS system.
3. Central system sends positions of selected objects (cars, houses, etc.) to mobile users. (Should be configurable at the central system.) The objects are displayed on a car GIS system.

Phase III – Integration of the query system part and the GIS part

[The resulting client application GUI could be two windows displayed at the same time: one for the messages part, the other for the GIS part. A query window may be an additional pop-up window at the top]

Architecture/hardware issues:

7. The system consists of a client application, an HTTP server and a DB server.
8. XML should be used for communication between the HTTP server and the client.
9. SQL should be used for communication between the DB server and the HTTP server. (DB server may, or may not, be able to deliver query result in XML.)
10. Intranet will probably be used (but the system should also work using the Internet).
11. The client program should be designed so that it could be used by a keyboard with minimum number of keys (perhaps only number and functional keys; letter keys could be displayed on the screen).
12. Tests will be done using a regular laptop. If the project goes well, EB may decide to buy a special car computer.

5. Meeting With ST & EB

Meeting notes
2002 09 24 @ ST
Participants: Giedrius Slivinskas, Andrius Dienys, Dario Pacino, Hjortur Scheving (ST), Alius Sadeckas (EB)

Demonstration of the current version

Comments from EB:

1. Each application should be in one window, occupying the whole screen.
2. Mobile user should see a log of his actions (e.g., ‘query sent’, ‘result received’, etc.; when the action is clicked, he/she should see the query or the result)
3. If central operator sends a query result to a mobile user, he/she should see if the result is received or not.
4. A progress bar with result percentages would be helpful.
5. Images can be downloaded after separate request.
6. Mobile-user and central-operator applications could be the same (if an authentification mechanism with different user roles is provided)

Software requirements specification

The specification will be further developed to address new requirements.
EB will send comments about the current version.

Next steps

1. Hardware – ST contacts Propac AB to see if it possible to rent a car computer
2. Further development of the system
    a. The query system (messaging)
    b. The GPS system (first demo version)
    c. Improvement of graphical user interface
    d. Connecting the system to the EB demo database
    e. User authentification
3. Demonstration strategy for Infobalt will be discussed later.

Preliminary date for the next demo: 2nd/3rd week of October (the demo should feature some of the functionality given in Next Step 2).

Next steps:

1. ST prepares a software requirement specification for the query system part (to be reviewed by EB).
2. ST prepares sketches for GUI (to be reviewed by EB).
3. ST starts developing the query system part
    a. Test with HTTP and DB servers at ST
    b. Test with HTTP and DB servers at EB

Notes for the query system part:

1. Example table with static objects can be used. 10 fields, one of them displays graphical information. The table structure is delivered as text file in SQL or other format (listing the fields and their types).
2. GUI should be as simple as possible.
3. The system should have authentification control. Mobile users should log into the system, and the operator at the central computer should see users who are online (and select them for sending information, see Stage I, Step 4 or Stage II, Step 3).

Appendix M

1. Bibliography

1. Object Oriented Analysis & Design
   Lars Mathiassen, Andreas Munk-Madsen, Peter Axel Nielsen, Jan Stage

2. Java™ Server & Servlets
   Rossback & Schireiber
   Published by Addison-Wesley
   ISBN 0-201-67491-2

3. Project Management for Information Systems, 3rd Edition
   James Cadle & Donald Yeates
   Published by Pearson Education
   ISBN 0-273-65145-5

4. The Java™ Programming Language, 3rd Edition
   Ken Arnold, James Gosling, David Holmes
   Published by Addison-Wesley
   ISBN 0-201-70433-1

5. Professional XML (Programmer to Programmer), 2nd Edition
   Mark Birbeck (Editor), Nikola Ozu, Jon Duckett, Andrew Watt, st Mohr, Oli Gauti Gudmundsson, Jon Duckett, Andrew Watt, Stephen Mohr, Kevin Williams, Oli Gauti Gudmundsson, Raja Mani, Daniel Marcus, Peter Kobak, Evan Lenz, Mark Birbeck, Brian Hickey, Zoran Zaev, Steven Livingstone, Jonathan Pinnock, Keith Visco
   Published by Wrox Press Inc
   ISBN 1-861-00505-9

6. The Rational Unified Process: An Introduction (2nd Edition)
   Philippe Kruchten
   Published by Addison-Wesley Pub Co
   ISBN 0-201-70710-1

7. Extreme Programming Explained: Embrace Change
   Kent Beck
   Published by Addison-Wesley Pub Co
   ISBN 0-201-61641-6

8. Planning Extreme Programming
   Kent Beck, Martin Fowler
   Addison-Wesley Pub Co
   ISBN 0-201-71091-9

2. Webography

1. ASP Resource Index
   http://www.aspin.com/

2. About GPRS
   http://www.ericsson.com/mobilityworld/sub/open/infrastructure/gprs/index.html?PU=gprs

3. Network Programming with J2ME Wireless Devices
   http://www.wirelessdevnet.com/channels/Java™/features/j2me_http.phtml

4. Extensible Markup Language
   http://www.xmlfiles.com/

5. Java™ Resources
   http://Java™.sun.com
   http://www.Java™almanac.com

6. Methodology recourses
   http://www.rational.com
   http://www.agilemodeling.com/essays/agileModelingRUP®.htm
   http://www.therationaledge.com/
   http://www.greenmountain.nu/
   http://members.aol.com/humansandt/crystal/clear/
   http://www.softwarereality.com/lifecycle/xp/
   http://www.martinfowler.com/articles/newMethodology.html
   http://www.agilemanifesto.org
   http://www.extremeprogramming.org

3. Documents and White Papers

1. Lessons Learned Practicing Agile Development
   Uttam Narsu and Liz Barnett

2. The Rational Approach to Automated Testing
   A Rational Software White Paper

3. Applying Requirements Management with Use Cases
   Roger Oberg, Leslee Probasco, and Maria Ericsson

4. Tailoring the Rational Unified Process, a Lightweight Process Development Case
   ICONIX Software Engineering, Inc. and General Information Systems Technology, Inc.

5. Use Case Management with Rational Rose® and Rational RequisitePro
   A Rational Software White Paper

6. Getting the Most Out of an Automated Test Tool
   Laura Lee Rose

7. RUP®/XP Guidelines: Pair Programming
   A Rational Software White Paper

8. Using the Rational Unified Process for Small Projects: Expanding Upon eXtreme Programming
   A Rational Software White Paper

9. RUP® Framework, support documents and tools
   Rational Software Corporation

10. RUP®/XP Guidelines: Test-first Design and Refactoring
    Robert C. Martin
    Object Mentor Inc.
    Rational Software White Paper

11. Avoiding the Death Spiral in Software Development
    Steve MacCarthy, Freddie Mac

12. A Comparison of RUP® and XP
    John Smith
    Rational Software White Paper