# SPITS in vehicle koppelingsprotocol voor een mobiel apparaat





**Afstudeerverslag**


| | |
|---|---|
| Auteur | : Jessie J.J. Hernandez |
| Studentennummer | : 20050537 |
| Datum | : 24- september 2010 |
| Versie | : 1.0 |
| Afstudeerbedrijf | : Logica Nederland BV |
| Bedrijfsbegeleiders | : Bert de Neef, Edwin Essenius |
| Bedrijfsmentor | : Richard Spruit |
| Onderwijsinstelling | : Haagse Hogeschool |
| Opleiding | : Technische Informatica |
| Examinator | : Hans Witkamp, Cobie van der Hoek |

# Referaat

Jessie Hernandez, afstudeerverslag, Spits in vehicle koppelings protocol, ’s-Gravenhage, Haagse Hogeschool, acedemie voor ICT & Media, Logica, Rotterdam, 24 september 2010.

**Samenvatting**

Dit verslag heeft betrekking op de opdracht “Spits in-vehicle koppelingsprotocol”. De opdracht is uitgevoerd bij Logica te Rotterdam in de Working Tomorrow afdeling in samenwerking met het Spits project. Dit project is tevens ook de laatste stap in het traject van de opleiding Technische Informatica van de Haagse Hogeschool.

**Descriptoren**

- Android
- USB
- ATOP
- Serieel
- Linux kernel
- Java
- Working Tomorrow
- Spits

# Voorwoord

Voor u ligt mijn afstudeerverslag van mijn afstudeerproject “Spits in vehicle koppelingsprotocol”. Deze heb ik geschreven in het kader van de opleiding Technische Informatica aan de Haagse Hogeschool. Dit project is uitgevoerd bij Logica in het Working Tomorrow afstudeerprogramma te Rotterdam.

Bij deze wil ik de mensen bedanken die mij geholpen hebben tijdens mijn afstudeerperiode met uitvoering van de opdracht, feedback op de documenten en ondersteuning waar het nodig was.

Ten eerste wil ik mijn begeleiders Bert de Neef, Projectleider Working Tomorrow, Edwin Essenius technische begeleider Working Tomorrow en het gehele Spits team te Rotterdam en Eindhoven bedanken voor de ondersteuning die jullie allemaal geboden hebben.

Verder wil ik ook mijn school begeleiders Hans Witkamp en Cobie van der Hoek bedanken voor de steun en begeleiding vanuit de academie

En als laatste mijn lieve ouders Henry en Lucille Hernandez, mijn twee broertjes Kristian en Stephan Hernandez en niet te vergeten mijn vriendin Stephanie Boekhouwer voor alle steun die ik gedurende de jaren heen heb gehad. Zonder jullie zou het niet gelukt zijn om deze belangrijke mijlpaal te halen.

's-Gravenhage, 24 september 2010
Jessie J.J. Hernandez

# Inhoud

# Inleiding

Spits is het open platform voor intelligente applicaties in de auto. Maar wat betekent dat precies? Stel je voor je naar je werk moet, in tegenstelling tot normaal word je vandaag geholpen door Spits. Spits maakt een open platform voor de communicatie tussen auto's, de weginfrastructuur en remote services om je mobiliteit, veiligheid en gemak te verhogen.

Aankomende bij een drukke parkeergarage word je door spits geleid naar een leeg parkeervak. Om bij te houden welke parkeervakken beschikbaar zijn moeten deze gegevens centraal worden opgeslagen en bijgehouden. Deze centrale plaats wordt de BackOffice genoemd.

Om de communicatie met de BackOffice te maken wordt er gebruik gemaakt van een mobiele internetconnectie.

Het mobiele internet is tegenwoordig bijna niet meer weg te denken uit het dagelijks leven. Inmiddels is het al mogelijk om via het mobiele internet: internetradio af te luisteren, E-mails te verzenden en video's te streamen. Om meer gebruik te maken van je mobiele internetconnectie is het nu mogelijk om je internetconnectie te gebruiken als modem voor je Spits OBU. Dit project gaat over het bouwen van een communicatielink tussen een Spits On Board Unit en een mobiel toestel met internet mogelijkheden.

Spits is een vervolg project op het CVIS project. CVIS staat voor Cooperative vehicle-infrastructure systems en is een internationaal project met als doel een geüniformeerde technische oplossingen te bedenken voor transparante communicatie tussen voertuigen en de infrastructuur door een groot aantal media te gebruiken.

mijn project is uitgevoerd bij het Working Tomorrow programma binnen Logica te Rotterdam. Meer informatie over Logica is te vinden in hoofdstuk 1.

In hoofdstuk 2 wordt het Spits project ingeleid. In dit hoofdstuk wordt uitgelegd waar de Spits architectuur uit bestaat. Verder wordt uitgelegd waarom ik gekozen heb voor de software ontwikkelmethode RUP.

In de inceptiefase ga ik mijn project definiëren. Dit houdt in dat ik de probleemstelling en de doelstelling van mijn project ga formuleren. Om aan de eisen van RUP te voldoen ga ik ervoor zorgen dat aan het eind van mijn inceptiefase een overeenstemming is tussen alle belanghebbenden. De inceptiefase zal ik verder toelichten in hoofdstuk 3

In hoofdstuk 4 ga ik mijn elaboratiefase uitwerken. In de elaboratiefase ga ik de eisen van de communicatielaag opstellen en uitzoeken hoe USB achter de schermen werkt. De scope van mijn project ga ik ook in mijn elaboratiefase beschrijven.

Ik ga in de constructiefase werken met twee iteraties. Ik kies hiervoor om het programmeerwerk overzichtelijk te houden. Doordat ik op meerdere platformen moet programmeren is het kiezen van iteraties een zeer aangename ervaring. De constructiefase ga ik in hoofdstuk 5 beschrijven.

Ter afsluiting van mijn project zal ik in hoofdstuk 6 een evaluatie schrijven. In de evaluatie worden ook de aanbevelingen gegeven. Verder worden ook enkele punten genoemd voor eventuele toekomstige projecten.

# 1 Organisatie

## 1.1 Logica

Logica is een belangrijke internationale speler op het gebied van IT en business services met 39.000 mensen in dienst in 36 landen. Ze bieden oplossingen en diensten in het domein van consultancy, design, systeem integratie, bedrijfskritische applicaties en de outsourcing van bedrijfsprocessen. Logica richt zich op vier marktsectoren – nutsbedrijven en telecom , financiën, de publieke sector, industrie distributie en Transport. Door deze focus zijn ze in staat oplossingen op maat te bieden voor de uitdagingen waar klanten in hun specifieke markt mee te maken krijgen. In Nederland is deze focus verankerd in view Industrie Sectoren, afdelingen die zich richten op een van de segmenten.

Logica is gedreven op klanten te helpen om leidende posities te behalen en te behouden in hun individuele marken. De kracht van Logica ligt daarbij op het gebied van industrie- en Domeinkennis, de sterke bedrijfskundig- en technologisch inzicht, en de vele successen met het op tijd en binnen budget leveren.

Het huidige Logica PLC. is op 30 december 2002 ontstaan uit het voormalige Logica PLC. (60%) en het voormalige CMG PLC. (40%). CMG was in Nederland aanzienlijk groter dan Logica, die vooral een focus had in Groot Brittannië. In 2006 zijn daar nog WM-Data (zwaartepunt in Scandinavië) en Unilog (zwaartepunt in Frankrijk) aan toegevoegd, zodat een gespreide dekking in Europa verkregen is. In 2008 is onder leiding van de nieuwe CEO, Andy Green, een programma opgestart om van alle onderdelen binnen het bedrijf een uniforme organisatie neer te zetten - One Logica. Hierbij zijn de waarden gedefinieerd, volgens welke iedere Logicaan zou moeten werken: betrokken, innovatief en open. De missie die hierbij gesteld is, is “To be the most trusted innovation partner”.

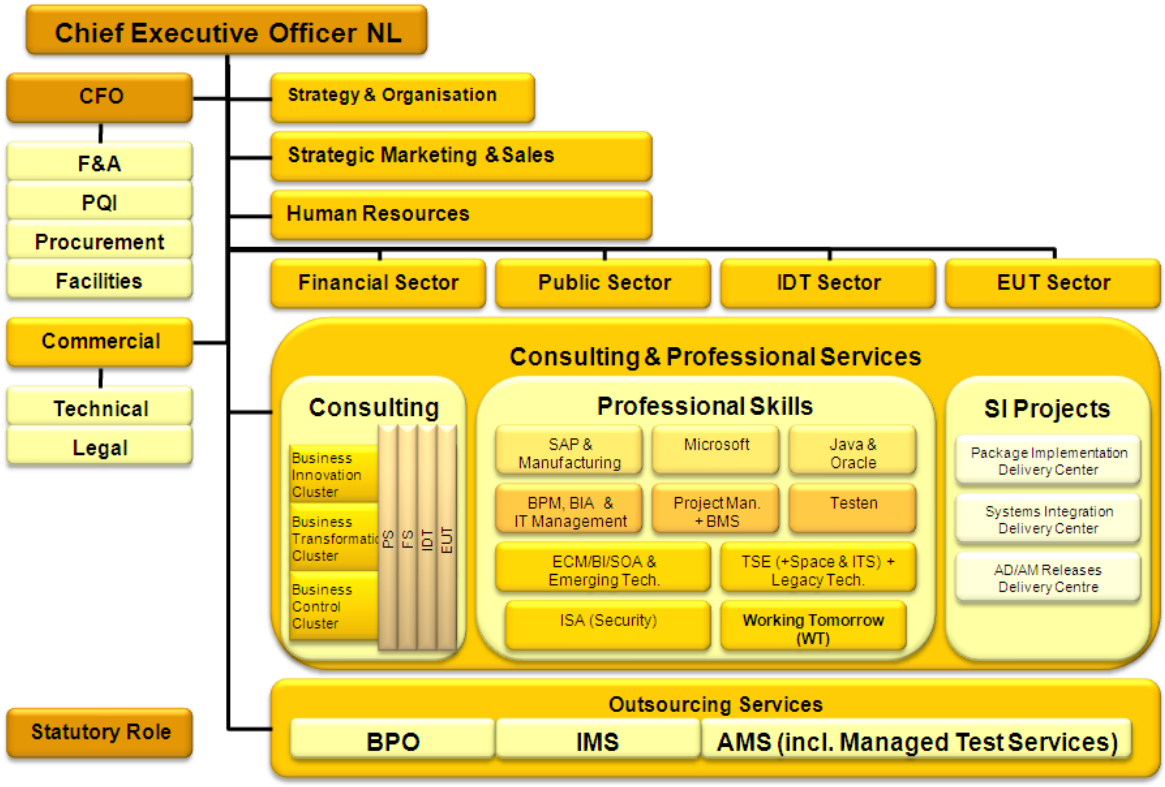
## 1.2 Working Tomorrow

De opdracht wordt uitgevoerd binnen het programma Working Tomorrow. Logica heeft dit programma opgestart om studenten de gelegenheid te geven af te studeren op een innovatieve opdracht met goede begeleiding. Innovatie staat centraal. Innovatief wat betreft technologie, concept of methodiek. Zo werken er studenten aan agenttechnologie, Multitouch technologieën, Augmented Reality, Smart Grids en slimme meters en mobiele oplossingen. Working Tomorrow is nu op 6 vestigingen van Logica in Nederland aanwezig en biedt jaarlijks plaats aan circa 120 studenten van HBO en WO. Op iedere vestiging is een projectleider van Working Tomorrow die de studenten begeleidt.

Het Working Tomorrow programma heeft 4 hoofddoelen:

1. Een centrale plaats bieden waar studenten uitdagende en innovatieve afstudeerprojecten kunnen uitvoeren;
2. Het werven van toekomstige werknemers;
3. Verhogen van de reputatie van Logica op het gebied van innovatie;
4. Demo’s en resultaten van projecten gebruiken voor het verkrijgen van betaalde opdrachten en vergaren van kennis.

Logica Nederland heeft naast de Industrie Sectoren ook Service Lines (SL). Consulting en Professional Services (C&PS) is er op gericht om de IS zo goed mogelijk te voorzien van goed en gekwalificeerd personeel. C&PS is verder onder te verdelen in Consulting, SI Projects en Professional Skills. Professional Skills is weer verder onderverdeeld in practices, dit zijn afdelingen die gericht zijn op een bepaalde skillset, bijv Project Management, Java.Oracle of ISA. Alle studenten die bij WT afstuderen zijn daarnaast ook ondergebracht in een practice die nauw aansluit bij hun opdracht.




**Figuur 1: Organogram Logica Working Tomorrow**

Als student die afstudeert bij Working Tomorrow, word ik bij een practice geplaatst die bij de richting van mijn project past. Ik krijg naast een procesmatige begeleiding en technische begeleiding vanuit Working Tomorrow ook inhoudelijke begeleiding vanuit de practice. De practice waar ik onder val is Technical Software Engineering.

## 1.3 Technical Software Engineering

Technical Software Engineering (TSE) is de software bouw afdeling van Logica. TSE is gespecialiseerd in diensten en projecten die een technische focus hebben. Technologie wordt steeds complexer. Daarom moeten de systemen waar deze technologie in is verwerkt, simpeler en makkelijker in gebruik worden. Veel apparaten worden ook “slimmer”, denk maar aan auto's, huishoudelijke toepassingen en productiemachines. Software schrijven voor deze apparaten is een gespecialiseerde taak en dit is het domein van TSE.

## 1.4 Begeleiding

Logica heeft per locatie ervaren professionals vrijgemaakt om de student bij de afstudeeropdracht te begeleiden. Ik heb een mentor toegewezen gekregen die mij wegwijs heeft gemaakt in de organisatie zodat ik me tevens goed kan voorbereiden op mijn verdere loopbaan. Naast deze directe begeleiding heb ik ook veel contact gehad met specialisten die bij het Spits project betrokken zijn. De werksfeer is informeel en betrokken; we helpen elkaar om onze doelstellingen te bereiken. Logica heeft bewust voor deze aanpak gekozen: Logica doet er alles aan om mij een gedegen oriëntatie op een professionele carrière te bieden. Of dat nu binnen Logica is of ergens anders; Logica vindt het belangrijk dat die beslissing gefundeerd en gemotiveerd genomen moet worden.

# 2 Wat is Spits

In dit hoofdstuk wordt het Spits project geïntroduceerd. De verschillende onderdelen van de Spits architectuur worden onderhanden genomen en uitgelegd. Deze kennis is nodig om mijn afstudeerproject binnen Spits te kunnen plaatsen. Na de introductie volgt een beschrijving van mijn keuze voor de software ontwikkelmethode RUP.

Op de website van Spits staat het volgende over Spits:

Spits is een Nederlands project belast met het ontwerpen van Intelligent Traffic Systems (ITS); concepten die de mobiliteit en veiligheid kunnen verbeteren.

Het Spits project richt zich op drie belangrijke gebieden:

1. Het beheren van verkeer blijft een relevant onderwerp voor ons allemaal. Onze wegen worden steeds voller en er is een toenemende behoefte om oplossingen te vinden om het verkeer in beweging te houden. ITS geeft de mogelijkheid om het bestaande wegennetwerk efficiënter te gebruiken en het verhogen van de totale bruikbaarheid van de wegen. Betere informatie kan ook de veiligheid op de wegen verhogen, bijvoorbeeld bij het naderen van scherpe bochten of het waarschuwen van andere weggebruikers over aankomend dreigend gevaar. Het Spits project zorgt voor het definiëren van een open en schaalbaar platform voor systemen in de toekomst en het verkennen van nieuwe technieken in coöperatie en mobiliteit Op die manier kan Logica bijdragen aan een efficiëntere en duurzame mobiliteitsoplossing.
2. In-vehicle oplossingen zullen een link leggen tussen een voertuig en de buitenwereld. Zoals de vorderingen in de in-vehicle telematicabranche zich ontwikkelen zien wij deze verkeersmanagement apparaten als een logisch vervolg op het in-vehicle infotainment systeem. In het Spits project wordt een open en uitbreidbaar systeem onderzocht, zodat de modernste producten altijd kunnen worden uitgerold. Het tempo van moderne innovatie betekent dat verschillende generaties van technologie worden uitgebracht tijdens de levensduur van een voertuig. Dus een upgradebare benadering is van essentieel belang om in-vehicle verkeersystemen bij de tijd te houden.
3. Een service downloaden en managementoplossing. Verkeersmanagementsystemen en up to date verkeersinformatie is de laatste tijd erg populair geworden. Spits onderzoekt de mogelijkheden van een service download en management oplossing voor in-vehicle diensten die bruikbaar zijn voor de consument, vloot en verkeersapplicaties. Naast de mogelijkheid van hardware upgraden zal de software die op deze systemen draait ook evolueren, net als de data die er gebruikt wordt. Door een simplistisch en open benadering te hanteren zal dit de innovatie en snelheid verhogen, zodat er snel nieuwe en spannende diensten kunnen worden ontwikkeld die het verkeersmanagement makkelijker maken en de veiligheid van de bestuurder verhoogt.

Spits wordt gefinancierd door dertien partners en het Ministerie van Economische Zaken.[1]

Binnen de SPITS architectuur zijn er verschillende applicaties met verschillende communicatie behoeftes. Hierdoor kunnen de applicaties in vijf verschillende groepen verdeeld worden.

Applicaties die gebruik maken van een connectie tussen:

1. Een auto en de BackOffice;
2. Autos onderling;
3. De weginfrastructuur onderling;
4. Een auto en de weginfrastructuur;
5. De verschillende BackOffices onderling

U kunt een afbeelding terugvinden in de bijlage 2.

Om deze communicatiebehoeftes te vervullen, hebben de verschillende applicaties binnen de Spits architectuur elk behoefte aan verschillende communicatiekanalen. Bijvoorbeeld de BackOffice heeft een internetconnectie en eventueel een sms-server nodig, maar heeft geen behoefte aan een GPS kanaal terwijl een auto deze wel goed kan gebruiken. Ik zal alleen de elementen van de Spits architectuur beschreven die relevant zijn voor mijn project. Voor meer informatie over de Spits architectuur kunt u terecht op de website https://spits-project.com.

## 2.1 On Board Unit

De On Board Unit (OBU) is het onderdeel van de Spits architectuur die in de auto zit. Op de Spits website staat het volgende over de OBU:

De OBU wordt onderverdeeld in 3 productlijnen. Eén voor de high-end markt, deze zijn gericht op commerciële doeleinden, één voor de low-end markt, deze is ideaal voor alle auto's dus massa productie en één voor onderzoeksdoeleinden. De OBU voor onderzoeksdoeleinden zal de bestaande CVIS OBU gebruiken die aan een Tom-Tom navigatie apparaat wordt gekoppeld.[2]

### ATOP

Binnen het Spits platform zijn er verschillende soorten OBU's. Voor mijn afstudeerproject is de OBU van de chipfabrikant NXP beschikbaar gesteld. Deze OBU heet Automotive Telematics Onboard-unit Platform (ATOP). Op de website van NXP wordt het volgende over de ATOP beschreven.

De ATOP geeft de mogelijkheid om als een afzonderlijke OBU in een embedded systeem voor te komen waar verschillende applicaties in voorkomen. De ATOP kan ook gebruikt worden in meer geavanceerde diensten voor telematica platformen bestaande uit een groot aantal diensten en applicaties. Deze applicaties helpen de rijpatronen te verbeteren, welke op zijn beurt de brandstof consumptie en de $CO^2$ emissie minimaal houdt.

#### Flexibele connectiviteit

ATOP biedt marktleidende GSM prestaties en ultra lage stroom gebruik. Het geïntegreerde geheugen biedt geavanceerde functies zoals veilige over-the-air-upgrades. De GPS ontvanger biedt nauwkeurigheid en gevoeligheid die nog ongekend zijn in de auto-industrie. De op een ARM7 gebaseerde microcontroller biedt een groot aantal interfaces voor het eenvoudig koppelen van de ATOP in een auto. Er wordt gebruik gemaakt van interfaces zoals CAN, USB, meerdere seriële inter-processor bussen en A/D en D/A converters en een analoge in- en out-put voor audiosignalen.

### Open en flexibele multi-applicatie omgeving

Voor de business logica en dienstenaspecten van het ontwerp is er voor een open op Java gebaseerde multi-applicatie software omgeving gekozen die interoperabiliteit en veilige co-existentie in systemen met meerdere telematica toepassingen garandeert. De ARM7 omgeving die C en C++ gebruikt om de JAVA omgeving te voorzien met een laag niveau en echte real-time mogelijkheden voor applicatie integratie.

### veiligheid

de SmartMX-controllers bieden een hoog veiligheidsniveau voor een multi-applicatie omgeving. De controllers zijn hoofdzakelijk ontworpen voor een hoge veiligheid die meerdere interface opties hebben en de controllers zijn fraudebestendig. [3]

In figuur 2 staat de ATOP afgebeeld. In bijlage 1 wordt de ATOP afgebeeld inclusief de technische details van de ATOP.


**Figuur 2: ATOP (voor- en achterkant)**

## 2.2 BackOffice

De BackOffice (BO) is het gedeelte van de architectuur waar de data wordt verwerkt. Spits heeft de BackOffice als volgt gedefinieerd:

De BackOffice fungeert als een dienstencenter in de Spits architectuur door front-end en back-end diensten te leveren.

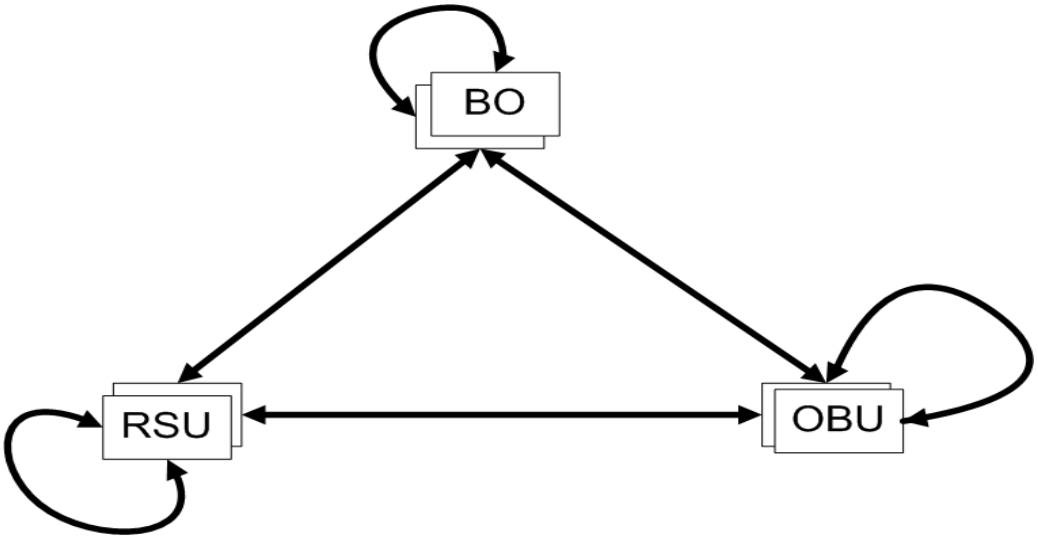
De BackOffice levert Front-end diensten aan andere belangrijke subsystemen van de Spits architectuur. Een elementaire front-end dienst van de BackOffice is het in staat stellen van de subsystemen zodat deze door een applicatie repository, applicaties kunnen downloaden. Deze applicaties zullen dan op de OBU's en RSU's[2] worden uitgevoerd. De BackOffice zal dan in staat zijn om een aantal van front-end diensten te hosten. Er zullen een aantal generieke diensten en een aantal diensten geleverd door derde partijen op de BackOffice draaien. Voorbeelden zijn bijvoorbeeld verkeersinformatie zoals waarschuwingen voor files, werkzaamheden en spookrijders, eventuele toeristische attracties in de buurt, reserveren van een parkeervak en tolheffingen.

Back-end diensten zorgen ervoor dat derden en beheerders hun diensten kunnen leveren en beheren. Bijvoorbeeld transacties beheren, roadtrips plannen, versies updaten. De BackOffice moet schaalbaar zijn.[4]

[2] RSU staat voor Road Side Unit en is deel van de Spits architectuur. De werking van de RSU valt buiten de scope van mijn project.

## 2.3 Het grote plaatje

Hieronder ziet u het grote plaatje van de Spits Architectuur. In de afbeelding ziet u hoe de OBU's, RSU's en de BO's met elkaar communiceren. In bijlage 2 wordt deze afbeelding in meer detail weergegeven met de communicatie protocollen die tussen de verschillende subsystemen van de Spits architectuur worden gebruikt.




**Figuur 3: Spits concept (Grote plaatje)**

## 2.4 Methode

Om mijn software ontwikkeltraject vorm te geven, ga ik een methode kiezen die het beste bij mijn project past. Ik ga hiervoor criteria opstellen om deze tegen een aantal methodes te testen. Omdat ik weinig tijd wil verliezen met het inlezen en begrijpen van een nieuwe methode kies ik ervoor om te focussen op methodes waar ik persoonlijk ervaring mee heb. Verder ga ik een methode kiezen die gebruikmaakt van een Iteratieve aanpak, dit is handig om overzicht te houden over het project. Andere criteria waar de methode aan moet voldoen is dat de verschillende fasen elkaar mogen overlappen. Mijn project heeft veel risico's die een grote impact op de uitkomst van mijn project kunnen invloeden hierdoor heb ik als laatste criteria bepaald dat de methode die ik ga kiezen risico's managent. In de tabel hieronder ziet u een vergelijking van een aantal methodes waar ik me keuze uit heb gemaakt. De methode met het beste risicomanagement en kleinste persoonlijke leercurve is RUP.

| Methode | Flexibele indeling | Iteratieve aanpak | risicomanagement | leercurve |
|---|---|---|---|---|
| RUP | Ja | Ja | Ja | Laag |
| Waterval | Nee | Nee | Nee | Middel |
| DSDM[3] | Ja | Ja | Ja | Hoog |

[3] Dynamic Systems Development Method

# 3 Inceptiefase

Volgens het boek "The Rational Unified Process An Introduction" van Philippe Kruchten wordt de inceptiefase als volgt beschreven "The overriding goal of the inception phase is to achieve concurrence among all stakeholders on the lifecycle objectives for the project."[5] Dit betekent dat het doel van de inceptiefase overeenstemming moet brengen tussen alle betrokken partijen.

RUP beschrijft voor elke discipline een aantal rollen. Ik zal de rollen die van toepassing zijn op mijn project opsommen en benoemen. Doordat ik mijn project alleen uitvoer, worden alle rollen door mij vervuld tenzij anders vermeld.

De project Managementdiscipline houdt in dat concurrerende doelstellingen worden gebalanceerd, risico's worden beheerd en beperkingen worden overwonnen om een product te leveren dat aan de eisen en wensen van de klant voldoet. In deze discipline is er een grote rol voor de project manager. Verder is er ook een rol voor de project reviewer. De project reviewer is verantwoordelijk voor de evaluatie van het project. De rol voor project reviewer wordt vervult door mijn begeleiders van school, Working Tomorrow en Spits.[5] Door elke week aan het eind van de week naar mijn voortgang te kijken kan ik bepalen of ik nog op schema lig en dit eventueel aan de opdrachtgever te melden.

Het doel van de disciplineeisen is het vaststellen en behouden van een overeenkomst tussen alle belanghebbenden over de eisen en wensen van het systeem. Deze discipline zorgt er ook voor dat de systeemontwikkelaars een beter beeld krijgen van wat er precies gebouwd moet worden. In de disciplineeisen zijn er twee rollen die vervuld moeten worden: De systeem analist en De eisenspecificeerder. De systeemanalist is verantwoordelijk voor het coördineren van de eisen en wensen. De systeemanalist zorgt er ook voor dat er use-cases gemaakt worden die de scope van het systeem weergeven. De eisenspecificeerder beschrijft de eisen en wensen die uit de use-cases naar voren komen.[5]. De rol van systeem analist vervul ik door de belangrijkste eisen en wensen van de opdrachtgever boven water te krijgen. Als eisenspecificeerder bespreek ik deze eisen met de opdrachtgever en vul deze zonodig aan.

De discipline analyse en ontwerp worden de eisen en wensen vertaald in een specificatie die beschrijft hoe het systeem moet worden geïmplementeerd. In de analyse en ontwerp discipline worden enkele rollen geïdentificeerd. De software architect is verantwoordelijk voor alle technische werkzaamheden en technische documenten die gedurende de levenscyclus van het project worden gemaakt. Een tweede rol in de analyse en ontwerp discipline is de ontwerper. De ontwerper is verantwoordelijk voor het ontwerpen van het gehele systeem.[5]. Als ontwerper ga ik de communicatielink ontwerpen en als software architect zal ik dan deze werkzaamheden controleren.

In de implementatiediscipline wordt het systeem gebouwd. Dit houdt in dat de broncode geschreven wordt. De rollen die in de implementatiediscipline worden beschreven zijn de implementator en de integrator. De implementator zorgt dat de code geschreven wordt en de integrator zorgt dat het systeem gecompileerd kan worden. Een andere rol in deze discipline is de code reviewer. Deze rol zorgt ervoor dat de code gecontroleerd wordt op kwaliteit. De rol van code reviewer wordt vervuld door mijn begeleiders van de Haagse Hogeschool en Logica.[5].

Ik zal de rollen code implementator en integrator samenvoegen tot één rol. En bij het schrijven van de code zal ik ervoor zorgen dat de code geschreven en tegelijkertijd ook gecompileerd kan worden.

De test discipline treedt op als een dienstenverlener voor de andere disciplines. De test discipline evalueert of het product wat opgeleverd is conform de afgesproken kwaliteit is. De test discipline bestaat uit vier rollen. De Test manager, de test analist, de test ontwerper en de tester. De testmanager is verantwoordelijk voor alle tests die uitgevoerd moeten worden. De testanalist is verantwoordelijk voor het identificeren en definiëren van de tests. De testontwerper is verantwoordelijk voor het definiëren van de aanpak die gebruikt wordt bij het testen en zorgt ervoor dat deze succesvol wordt geïmplementeerd. De tester is verantwoordelijk voor het uitvoeren van de tests.[5]. Als testmanager zal ik een testplan maken, voor de rollen testanalist en testontwerper zal ik de testgevallen beschrijven en bepalen en als tester de tests daadwerkelijk uitvoeren.

Configuratie en change management is de discipline die ervoor zorgt dat integriteit van alle documenten die geschreven zijn tijdens het project behouden blijft. De rollen die in deze discipline kunnen worden geïdentificeerd zijn de configuratiemanager en de changecontrol manager.[5]. Als configuratie- en changecontrol-manager ga ik geen complete configuratie managementplan schrijven. Ik werk alleen aan dit project dus alle belangrijke documenten zal ik lokaal bewaren en als back-up zal ik de diensten gebruiken van "Dropbox"[4].

De verantwoordelijkheid van de omgevingsdiscipline is het ondersteunen van de ontwikkeling van het systeem door tools te selecteren. De procestechnicus is verantwoordelijk voor het ontwikkel proces. Deze rol wordt door mijn Spits begeleider vervuld.[5]

De deploymentdiscipline zorgt ervoor dat de software wordt over gedragen aan de eindgebruikers. De rol in deze discipline is de deploymentmanager en die zorgt voor de uitrol van de software voor de eindgebruikers.[5]. Mijn eindproduct zal een Proof of Concept zijn en zal dus niet worden uitgerold.

## 3.1 Visiondocument

Het schrijven van een visiondocument helpt bij het komen tot een overeenstemming. In het visiondocument wordt een algemene visie voorgeschreven. In deze visie worden de belangrijkste eisen beschreven, de terminologie die gebruikt wordt tijdens het project wordt geïntroduceerd, een planning met iteratieplan, een risico analyse wordt beschreven, de scope en een mogelijke oplossing.[5]

Om de visie van mijn project te tonen is het schrijven van een visiondocument noodzakelijk. In mijn visiondocument worden de volgende punten behandeld: de aanleiding van mijn project, de probleemstelling, de doelstelling, de risico's, de randvoorwaarden, de verwachte output van mijn project en mijn planning en iteratiestrategie.

### 3.1.1 Aanleiding

De aanleiding van mijn project heb ik samen met mijn bedrijfsbegeleiders al in het voortraject van mijn afstudeeropdracht beschreven. Doordat het noodzakelijk is om vanuit mijn opleidingsinstelling al bij aanvang van mijn project een concreet plan te hebben. Op de aanleiding van mijn project was vrijwel geen commentaar. De aanleiding die ik heb opgesteld samen met mijn bedrijfsbegeleiders is als volgt:

[4] Dropbox is een online dienst waar je documenten kunt synchroniseren met verschillende Pc's. Verder heeft Dropbox ook de mogelijkheid om versies bij te houden.

Aanleiding voor dit project is de ontwikkeling binnen het SPITS project van conceptproducten voor derde partijen die diensten leveren voor autorijders.
Voorbeelden hiervan zijn verzekeraars, leasemaatschappijen, autoverhuurbedrijven en andere mogelijke wagenparkbeheerders.

SPITS wil oplossingen wegzetten waarmee dergelijke partijen diensten kunnen worden aangeboden zoals een remote-diagnostics functionaliteit (het op afstand monitoren van de toestand van een voertuig), een emergency-call functionaliteit (het automatisch inschakelen van hulpdiensten in geval van een ongeval), een stolen-vehicle functionaliteit (het tracken en tracen van gestolen auto's), of een pay-as-you-drive functionaliteit (het gebruik maken van het rijgedrag voor het bepalen van kosten zoals huurprijs, leaseprijs, of een verzekeringspremie).

Hierbij willen we gebruik maken van een vast kastje in de auto: de OBU (On Board Unit). De OBU is een in-vehicle device (of mogelijk combinatie van meerdere apparaten), die deel uitmaakt van het SPITS platform en waarop SPITS applicaties kunnen draaien.

Binnen SPITS wordt momenteel gewerkt aan de ontwikkeling van dergelijke in-vehicle systemen, die beschikken over standaard, vaste, communicatiekanalen, met de daarbij behorende communicatiekosten. De On Board Unit beschikt over mogelijkheden om te communiceren met de SPITS BackOffice, maar het is nog onduidelijk wat de kosten hiervan zullen zijn omdat de OBU nog een experimenteel product is.

**Citaat 1: Aanleiding (visiondocument)**

Doordat ik de aanleiding van mijn project al bij het schrijven van mijn afstudeerplan heb geschreven, kies ik ervoor om deze onveranderd te laten en direct over te nemen in mijn visiondocument.

### 3.1.2 Probleemstelling

Bij het bepalen van de probleemstelling zijn er wel enkele verbetermomenten aan bod gekomen. Net als bij het beschrijven van de aanleiding heb ik deze gedaan samen met mijn bedrijfsbegeleiders. In de eerste versie van mijn probleemstelling was niet duidelijk wat het probleem precies inhield. Het probleem hield in dat de opdrachtgever de mogelijkheid wil bieden om een OBU met de BackOffice te laten communiceren door gebruik te maken van de internet connectie van een mobiele apparaat. Hieronder ziet u de eerste versie van mijn probleemstelling

Voor een mogelijke partij, zoals bijvoorbeeld een verzekeringsmaatschappij, is de bruikbaarheid van, business case voor, en interesse in het gebruik van onze concept producten in belangrijke mate afhankelijk van de ermee verbonden kosten (zowel installatie als communicatie). Wat betreft de communicatiekosten zal een partij deze, afhankelijk van de betreffende applicatie, voor zijn eigen rekening willen nemen, of juist voor rekening van de gebruiker willen laten zijn. Zo zal een verzekeringsmaatschappij de communicatiekosten die verbonden zijn met een stolen-vehicle applicatie best voor eigen rekening willen nemen, maar de communicatiekosten verbonden met een pay-as-you-drive applicatie niet.

**Citaat 2: Probleemstelling versie 1**

Uit de formulering wordt het niet duidelijk wat het probleem precies inhoudt, doordat de probleemstelling nog te algemeen is. Bij het bepalen van een nieuwe probleemstelling zijn mijn begeleiders en ik tot de volgende probleemstelling gekomen.

Om onafhankelijk te kunnen zijn van een specifieke Telecom leverancier, willen we de mogelijkheid hebben om ook via een ander kanaal datacommunicatie te realiseren. Zonder gebruik te maken van de communicatie mogelijkheden van de OBU.

**Citaat 3: Probleemstelling (visiondocument)**

### 3.1.3 Doelstelling

De eerste versie van de doelstelling was helemaal in de verkeerde richting omdat ik bij de doelstelling al een oplossingsrichting beschrijf. Bij het bepalen van de doelstelling moet een situatie beschreven worden waarmee je aan het eind van het project kan meten of je project geslaagd is of niet. Hieronder ziet u de doelstelling zoals ik die heb opgezet in mijn eerste versie.

Het gebruik maken van een ander communicatiekanaal, waarvan de kosten voor de gebruiker zelf zijn, zou een oplossing kunnen zijn. Daarom willen we proberen om een smartphone (een zogenaamd "nomadic" device) te koppelen aan de een SPITS OBU, die voorzien is van een eigen communicatiekanaal, om het communicatiekanaal van de smartphone (3G) ter beschikking te stellen voor de OBU. De opdracht moet zich daarom richten op het mogelijk maken om mobiele apparaten, die beschikken over bijvoorbeeld goedkopere, andere, of reeds beschikbare data-communicatiekanalen te koppelen aan het SPITS in-vehicle systeem (OBU).

Hiervoor dient een applicatieprotocol te worden ontwikkeld waarmee het mogelijk wordt om mobiele apparaten eenvoudig en dynamisch te koppelen met de OBU, en de communicatie mogelijkheden van het apparaat ter beschikking te stellen voor het systeem. Het gaat dus om een stuk middleware, bestaande uit een software component op de OBU en een software component op de smartphone, waarmee applicaties die op de OBU draaien gebruik kunnen maken van de communicatiekanalen van een smartphone voor hun datatransmissie met de backend systemen van SPITS. Dit dient te gebeuren op een robuuste en flexibele manier; het wegvallen van een communicatiekanaal dient bijvoorbeeld goed (transparant voor de applicatie) te worden afgehandeld en het protocol moet voorzien in mogelijke uitbreidingen voor bijvoorbeeld andere functionaliteiten dan communicatie die kunnen worden geshared (bijvoorbeeld gebruik van een user interface). Tevens dient de goede werking ervan te worden aangetoond met een werkend prototype, bestaande uit een applicatie op de OBU de SPITS Back Office, gebruik makende van de ontwikkelde middleware op de OBU en de smartphone. Een vergelijkbaar stuk middleware is reeds eerder ontwikkeld in het CVIS project (Cooperative Vehicle Infrastucture Systems): het CALM concept (Communications Access for Land Mobiles). Echter, het gebruik van nomadic devices is hierin niet uitgewerkt en de ontwikkelde software draait op CVIS pc's en niet op de SPITS OBU's. De achterliggende gedachte van het CALM concept dient bestudeerd, geëvalueerd en meegenomen te worden in deze opdracht.

Citaat 4: Doelstelling versie 1

Ik heb de delen waar de oplossingsrichting wordt gegeven uit de doelstelling gehaald en heb de doelstelling als volgt opgezet. Na het verwijderen van de mogelijke oplossingen ben ik tot de volgende doelstelling gekomen. In deze doelstelling worden geen voorbeeld oplossingen voorgesteld maar puur het doel van mijn project beschreven. Hieronder ziet u de doelstelling zoals die in mijn afstudeerplan staat.

Het realiseren van een proof of concept van een middlewarelaag om datacommunicatie van de OBU met de back-office tot stand te brengen via een nomadic device[5].

**Citaat 5: Doelstelling (visiondocument)**

[5] Nomadic device is een mobiele apparaat met mobiel internet en heeft een USB poort.

### 3.1.4 Risico's

Om er zeker van te zijn dat ik gedurende mijn project niet vast kom te lopen door onvoorziene omstandigheden ga ik een risicoanalyse maken. Deze risicoanalyse bestaat uit de risico's waar ik tijdens mijn project tegen aan kan lopen, de kans dat deze voorkomen, de impact dat deze risico's op mijn project zullen hebben en een plan B. In de tabel hieronder ziet u de risicoanalyse zoals ik die in mijn visiondocument heb gemaakt.

| Risico | Kans | Gevolg | Maatregel |
|---|---|---|---|
| Ik heb geen nomadic device ter mijn beschikking | Klein | Groot | Ik kan mijn eigen nomadic device gebruiken |
| Ik heb geen auto met OBU om volledig invloeden van buiten te testen | Groot | Middel | Er wordt dan niet getest of er invloeden van buiten zijn bij de connectie tussen nomadic device en OBU |
| Ik ben langdurig ziek | Klein | Groot | Met de Hogeschool kan uitstel geregeld worden |
| Dataverlies | Klein | Groot | Elk document wordt op de netwerkschijf van Logica geplaatst. Als de gegeven ruimte vol is zal deze op een persoonlijke hardeschijf/flashdrive worden gedaan |

**Tabel 1: Risicoanalyse (visiondocument)**

De risico's ga ik bepalen door te kijken naar wat voor onvoorziene omstandigheden een negatief effect hebben op de uitkomst van mijn project. Hardware zoals een nomadic device is essentieel voor mijn project en betekent dus dat er rekening gehouden moet worden met de kans dat deze niet beschikbaar is. Deze risicoanalyse is gemaakt in de periode voordat ik ben begonnen aan mijn project hierdoor zijn er verschillende risico's die bij aanvang van mijn project niet meer van toepassing zijn. Het risico "geen nomadic device" is een van deze risico's die in de voorfase is opgenomen maar bij aanvang van mijn project is komen te vervallen. Een tweede risico dat is komen te vervallen is het risico "geen auto met OBU om invloeden van buiten te testen". Dit risico is bedacht toen het nog niet bekend was dat de OBU geen Bluetooth ondersteunt doordat de communicatielink nu alleen via USB werkt zijn invloeden van buiten niet meer van toepassing.

Verder zijn de standaard risico's zoals wanneer ik langdurig ziek ben of wanneer dataverlies optreedt. Deze risico's neem ik ook op anders worden ze over het hoofd gezien. Doordat ik bij aanvang van mijn project nog geen toegang tot het netwerk van Logica had was het niet mogelijk om mijn documenten op de netwerkschijf op te slaan. Als oplossing heb ik gekozen om de documenten online op te slaan met een dienst die alle documenten synchroniseert met andere computers en ook de mogelijkheid biedt om versies te kunnen beheren.

### 3.1.5 Randvoorwaarden

Bij elk project zijn er bepaalde voorwaarden waar een project minimaal aan moet voldoen om een kans te hebben om te kunnen slagen. Voor mijn project heb ik deze voorwaarden verdeeld in hardware en software eisen. Deze voorwaarden komen ook terug in de risicoanalyse die ik gemaakt heb. Hieronder ziet u de voorwaarden die ik gesteld heb aan mijn project.

hardware:

- een OBU
- een nomadic device met het Android besturingssysteem
- toegang tot de SPITS BackOffice

Software
Er is software nodig om het project succesvol te laten verlopen
- De software development kit van het Android besturingssysteem

**Citaat 6: Randvoorwaarden (visiondocument)**

Sommige risico's hebben een te grote impact op het project dat deze risico's gezien kunnen worden als randvoorwaarden. Om deze randvoorwaarden te bepalen heb ik gekeken naar de hard- en software die absoluut nodig is voor mijn project, de OBU bijvoorbeeld is van essentieel belang voor mijn project, als er geen OBU is kan ik mijn project niet uitvoeren.

### 3.1.6 Mogelijke Oplossing

Zoals ik in mijn doelstelling heb beschreven ga ik een middlewarelaag bouwen om de datacommunicatie van de OBU met de SPITS BackOffice tot stand te brengen via een nomadic device

De keuze voor de term "nomadic device" is tot stand gekomen omdat deze communicatielink niet alleen met verschillende smartphones moet kunnen werken maar ook met PDA's die een mobiele internetverbinding hebben, in theorie zou bijvoorbeeld de iPad gebruikt kunnen worden als nomadic device. De communicatielink zou ook met toekomstige mobiele apparaten moeten werken. Als nomadic device zijn verschillende apparaten mogelijk. De keuze bestaat uit bijvoorbeeld iPhone, Blackberry, PDA's en mobiele toestellen die op het Android platform draaien. Doordat er bij aanvang van mijn opdracht een Androidtoestel beschikbaar is gesteld ga ik de communicatielink bouwen op het Android platform.

### 3.1.7 Scope

In de literatuur wordt beschreven dat een scope aan het einde van het vision document gewenst is [5]. Doordat ik aan het begin van deze opdracht nog geen kennis over het domein bezit en vooral niet weet waarover Spits gaat, ga ik in de inceptiefase een gedeelte van de scope bepalen. Nadat ik me heb ingelezen zal ik de scope verder uitwerken dit zal ik in de elaboratiefase doen.
Het gedeelte van de scope die ik alvast in de inceptiefase ga definiëren zijn de hard- en software die ik ga gebruiken om mijn project uit te voeren. In de vorige paragraaf hebt u gelezen dat ik op het Androidplatform ga programmeren. Zoals het in de probleemstelling staat ga ik alleen werken met OBU's en BackOffices. Om deze reden zal ik geen rekening houden met de werking en functionaliteit van de RSU's. Om de applicaties te programmeren zal ik in de Eclipse IDE programmeren.

Het mobiele internet kanaal van de nomadic device wordt gebruikt om communicatie met de BackOffice te maken.

#### Nomadic Device

Zoals ik in hoofdstuk 3.1.6 heb beschreven is de nomadic device een Androidtoestel. Dit toestel is van het merk HTC Hero. De HTC Hero heeft een bedrijfseigen USB aansluiting die compatible is met een USB mini aansluiting, een Bluetooth 2.0 ontvanger met de EDR en A2DP profielen. De EDR profiel staat voor Enhanced Data Rate en wordt gebruikt voor data transmissie via Bluetooth. A2DP staat voor Advanced Audio Distribution Profile, dit profiel zorgt voor wat voor geluidskwaliteit er gebruikt wordt bij het initiëren van een bluetooth connectie met een headset. Verder heeft de HTC Hero een WIFI kanaal en biedt ondersteuning voor HSPA/UMTS/GSM/GPRS/EDGE. Als besturingssysteem draait er standaard Android versie 1.5 op. Voor de volledige technische details kunt u terecht op de website van HTC: http://www.htc.com/nl/product/hero/specification.html.

#### BackOffice

De Spits BackOffice is een server die via het internet bereikbaar is. Er is echter nog geen werkende Spits BackOffice. Ik zal om deze reden een simpele BackOffice programmeren die de data die vanuit de nomadic device verstuurd is ontvangt en toont.

## 3.2 Domein Onderzoek

Bij het inlezen heb ik een aantal documenten gekregen van mijn Spitsbegeleider met de kanttekening dat niet alle documenten relevant zijn voor mijn project. Om hiermee om te gaan heb ik voor mijzelf een selectieprocedure bepaald, om op een simpele wijze te beslissen welke documenten van belang zijn voor mijn project. Uit de gekregen documenten wil ik leren welke ideeën binnen Spits zijn over hoe de communicatie tussen elk element tot stand komt. Ik wil weten of er rekening gehouden moet worden met eventuele timing elementen tijdens het communiceren tussen de verschillende Spits subsystemen dit kan grote gevolgen hebben voor de manier waarop ontworpen en geprogrammeerd zal worden. Ik wil weten of er al mogelijke oplossingen binnen Spits zijn die ik eventueel zou kunnen uitwerken als dit zo is dan kan ik deze ideeën uitwerken en eventueel kijken naar mogelijkheden om het communicatiekanaal op andere platformen te bouwen.

De selectieprocedure die ik ga toepassen ziet er als volgt uit. Ik ga eerst op zoek naar documenten die te maken hebben met de communicatie protocollen die gebruikt worden binnen de Spits architectuur. Om deze reden ga ik documenten lezen die geschreven zijn voor het Spitsproject.

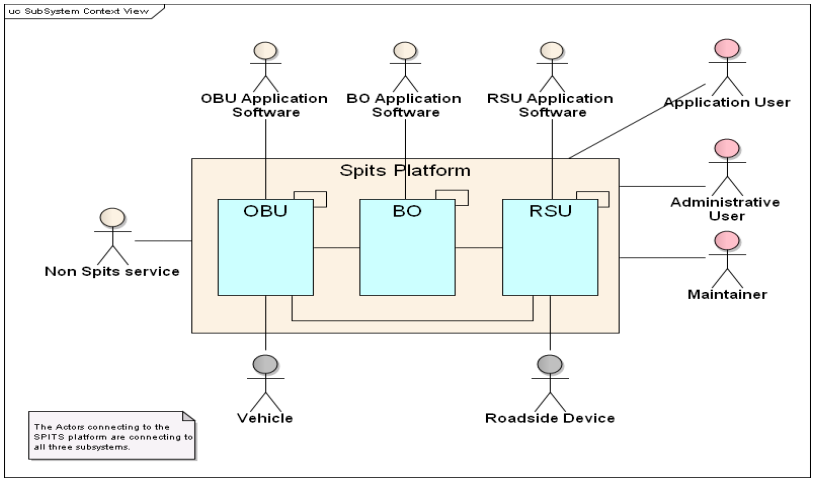
Uit de verkregen documenten zijn maar 3 documenten uit de 12 geschreven voor Spits. De rest van de documenten zijn geschreven voor het CVIS project. Doordat het Spitsproject veel overlap heeft met het CVIS project heb ik de rest van de documenten ook doorgenomen. Maar de documenten van Spits zal ik als belangrijkste beschouwen. De titels van de drie documenten geschreven voor spits zijn "Spits Network Architecture", “OverallArchitecture” en “802.11p approach”.

Deze documenten zien ernaar uit alsof ze informatie bevatten over de communicatie tussen de verschillende subsystemen binnen Spits en hoe het grote plaatje precies in elkaar steekt. Het lijkt me belangrijk om sowieso deze documenten aan de kant te leggen om uitgebreid door te nemen.

Hieronder volgt een samenvatting van de 3 belangrijkste documenten.

### 3.2.1 Spits Network Architecture

De netwerkarchitectuur van Spits moet ervoor zorgen dat de actoren met het platform kunnen communiceren. De netwerkarchitectuur moet ook de interne communicatie van de verschillende subsystemen van Spits onder handen nemen. In de afbeelding hieronder ziet u de verschillende belanghebbenden die gebruikmaken van de Spits netwerkarchitectuur. Deze afbeelding komt uit het document Spits Network Architecture v0.9.pdf[6]. In Bijlage 3 vindt u een grotere afbeelding.




**Figuur 4: Belanghebbenden bij het Spits Netwerk Architectuur**

In Figuur 4 kunt u zien dat er onderscheid is gebracht tussen de kleuren van de actoren. De roze actoren aan de rechterkant van de afbeelding zijn rollen die door mensen vervuld kunnen worden. Doordat dit echte mensen zijn, zullen ze geen netwerkinterface hebben. De andere actoren zijn apparaten of software componenten. De interface die nodig is om de OBU met de auto te verbinden is een bedrijfseigen interface en wordt niet verder behandeld. De non-spits service actor kan met alle drie de subsystemen communiceren dit kan via niet-netwerk interfaces (zoals GPS, FM Radio) en netwerk interfaces (zoals webservices). Een applicatie kan in 3 groepen worden verdeeld zoals in de vorige afbeelding is afgebeeld. De applicaties communiceren via API's met het Spits subsysteem. Applicaties kunnen ook onderling via het Spits platform communiceren.

Er zijn 21 scenario's bedacht om de Spitsarchitectuur te bepalen. Deze scenario's zijn niet verdeeld onder de subsystemen. De scenario's zijn bedacht op systeemniveau. Dit betekent dat er eerst de logische links tussen de subsystemen bedacht moest worden. Hieronder ziet u de logische links die mogelijk zijn.

- OBU – BO
- OBU - OBU
- OBU -RSU
- RSU – RSU
- RSU – BO
- BO- BO

Omdat het niet belangrijk is voor mijn project om aandacht te besteden aan elke logische link. Kies ik ervoor om alleen de logische link OBU-BO te onderzoeken, doordat deze van belang is voor mijn project. De logische link OBU-BO zal via het internet verbonden worden. De data die gebruik maakt van deze logische link zal unicast worden verzonden. Doordat de data via het internet verzonden wordt geeft dit de mogelijkheid om een keuze te maken voor IPv4 of IPv6.

In de huidige standaardisatie van coöperatieve systemen door ISO, ETSI, IEEE en CEN is IPv6 de standaard IP protocol voor Wannetwerken. Er zijn twee redenen hiervoor. Coöperatieve systemen zijn erg dynamisch. Dit betekent dat de netwerkconfiguratie ook erg dynamisch moet zijn. dit is vooral belangrijk voor korte afstand netwerkstandaarden zoals WiFi 802.11p. Deze standaard wordt waarschijnlijk gebruikt voor de logische link OBU-RSU. IPv4 is niet geschikt voor dynamische

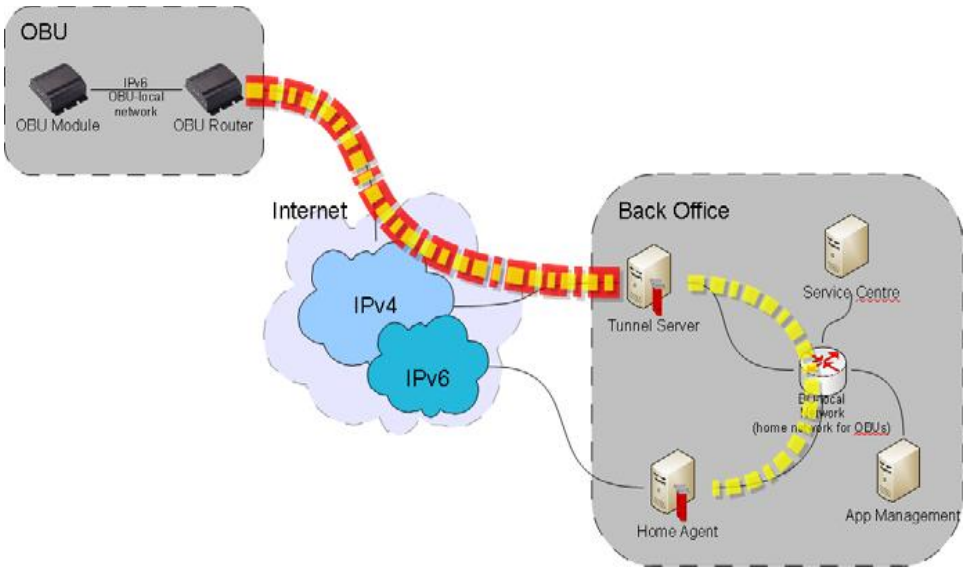
netwerken terwijl IPv6 hier wel tegen kan. De tweede reden voor het kiezen van IPv6 is dat IPv4 een tekort aan adressen heeft. Om coöperatieve systemen succesvol te implementeren heb je een significante marktpenetratie nodig. Het zal vrijwel onmogelijk zijn om genoeg IPv4 adressen te kopen tegen betaalbare prijzen om elke node van een IP te voorzien. Omdat Spits een open platform wilt ontwikkelen dat aan internationale standaarden voldoet wordt IPv6 gekozen.

De huidige stand van het internet biedt nog geen volledige IPv6-ondersteuning aan alle nodes, om deze reden moet Spits een oplossing bedenken om IPv4 als transport protocol te gebruiken voor IPv6. Dit kan worden bereikt door twee veranderingen te brengen aan de netwerktopologie.

1. De OBU maakt een connectie naar een tunnel server via OpenVPN of SSH. Dit voorziet de OBU van connectie binnen het thuisnetwerk binnen de BO.
2. De BO maakt een connectie met de Tunnel Broker die het netwerk van IPv6 adressen voorziet.

Om de OBU van een IPv6 adres te voorzien moeten er twee tunnels opgezet worden. Om dit uit te leggen ga ik uit van de volgende situatie.

1. Een tunnel wordt opgezet vanuit de OBU zijn router naar de Tunnel Server in de BackOffice. Deze tunnel is een IPv4 tunnel(zie figuur 5)
2. Door de reeds opgezette tunnel kan de OBU aan de Home Agent vertellen dat hij beschikbaar is voor het IPv6 netwerk. Daarna wordt een tweede tunnel opgezet tussen de OBU en de Home Agent om IPv6 pakketen te versturen.



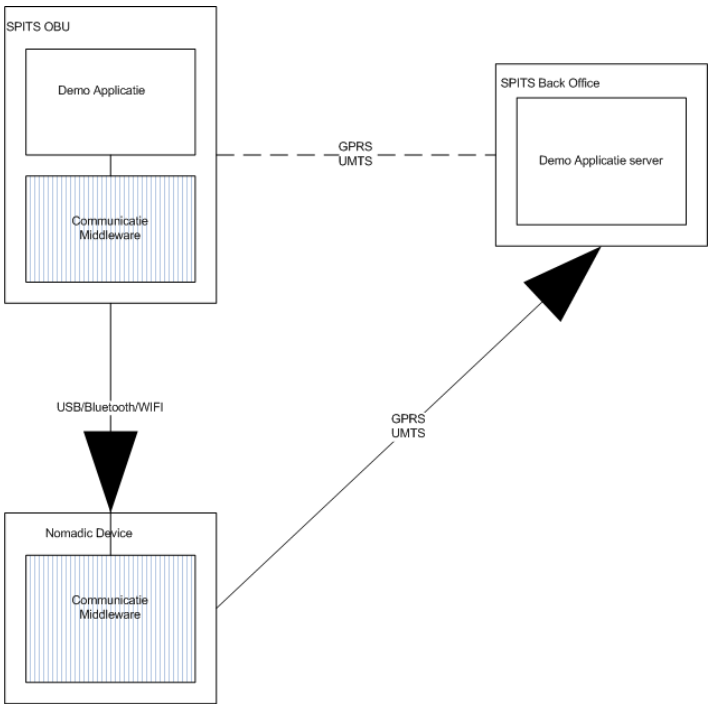

**Figuur 5: IPv6 tunnel**

Deze informatie is uit het document Spits Network architecture v0.9[6]

### 3.2.2 Globale oplossingsrichting

Bij het maken van mijn middleware laag is het niet van belang welke applicatie er op dat moment bezig is met het versturen van data. De reden hiervoor is doordat de middleware laag er alleen voor zorgt dat de data verstuurd wordt. De middleware laag is niet bedoeld om data te filteren.

Na het inlezen van de netwerkarchitectuur heb ik voor mezelf een schets gemaakt waar ik duidelijk wil maken wat het bereik van mijn project in een vroeg stadium is. Hieronder ziet u de schets die ik heb gemaakt. Voor een overzicht met meer detail kunt u kijken in bijlage 4




**Figuur 6: Schets van het probleem**

In deze schets wil ik weergeven hoe de datastroom verloopt wanneer de middleware laag geïmplementeerd is. De demoapplicatie gaat data verzenden en stuurt dit naar de Communicatie middleware, deze middleware stuurt de data door naar de nomadic device. De nomadic device zal dan aan de hand van de data die net binnen is gekomen bepalen naar welke BackOffice de verstuurd moet worden. Het versturen zal gebeuren via het GPRS/UMTS kanaal.

## 3.3 Planning

### 3.3.1 Globale Planning

Ik ga ook een planning maken om mijn project te beheren. Hiervoor heb ik voor mezelf bedacht waar en hoe ik in mijn project iteraties ga gebruiken. Dit heb ik gedaan door te kijken naar wat de doelstelling van de opdracht is en hiervoor zelf iteraties te bedenken. Bij het maken van mijn planning ben ervan uitgegaan dat een week vijf werkdagen en een dag acht werkuren inhoudt.

Voor de inceptiefase heb ik twee weken ingepland. Doordat ik bij het opzetten van mijn afstudeerplan al veel tijd heb gestoken in het bepalen van mijn plan van aanpak.

In de elaboratiefase heb ik gepland dat ik ga inlezen in de documentatie die beschikbaar is.

Verder wil ik ook gaan uitzoeken hoe USB en Bluetooth precies in elkaar zitten en wat er allemaal bij komt kijken om deze twee communicatiekanalen te programmeren. Verder in de planning staat er ook tijd uitgerekend om het architectuurrapport te schrijven. Hiervoor heb ik 5 weken uitgetrokken.

Doordat ik met twee systemen moet werken was het relatief makkelijk om alvast twee iteraties in de constructiefase te bepalen. Ik heb besloten om een iteratie te doen voor het bouwen van de communicatielink tussen nomadic device en BackOffice en een iteratie voor het bouwen van de communicatielink tussen OBU en nomadic device.

In eerste instantie was het de bedoeling om een communicatielink te bouwen met behulp van de communicatiekanalen USB en Bluetooth. Hierdoor is ook besloten om dan nog een iteratie in te plannen om dan de communicatie link voor het tweede communicatie kanaal te bouwen. Dus bij aanvang van mijn project waren er drie iteraties gepland. Een iteraties voor het bouwen van de communicatielink tussen OBU en nomadic device voor USB, een iteratie voor het bouwen van de communicatielink tussen nomadic device en BackOffice voor USB en een iteratie die de communicatielink voor Bluetooth implementeert. Doordat alleen de methode van connectie zou moeten veranderen hoef ik niet het gehele ontwerp proces doorlopen om nog eens de communicatielink voor Bluetooth te ontwikkelen. Voor elke iteratie heb ik 3 weken uitgetrokken.

Om toch nog een beetje speling in mijn planning te houden heb ik nog een extra week ingepland als eventuele uitloopweek. Deze week is ingepland voor eventuele feestdagen en/of vakantiedagen.

De planning zoals ik die bedacht heb bij aanvang van mijn project kunt u vinden in bijlage 5.

Nadat ik de planning heb gemaakt heb ik deze laten controleren door mijn schoolmentor en mijn bedrijfsmentoren. De controle op de planning was om te kijken of de planning realistisch was.

### 3.3.2 Mijlpalen

Om de planning overzichtelijk te houden heb ik alle belangrijke mijlpalen in een tabel gestopt. Dit maakt het bijhouden van welk product wanneer af moet zijn makkelijker. Om de mijlpalen te bepalen heb ik gekeken naar wanneer het betreffende product af moet zijn.

Hieronder ziet u de mijlpalen die ik heb bepaald en wanneer deze klaar zijn.

| **Mijlpaal** | **Deadline** |
|---|---|
| Inceptiefase af | |
| PvA wordt opgeleverd/ vissiondocument opgeleverd | 28 mei 2010 |
| Elaboratiefase | |
| Architectuur rapport opgeleverd | 25 juni 2010 |
| Constructiefase | |
| Constructie rapport (iteratie 1) opgeleverd | 16 july 2010 |
| Constructie rapport (iteratie 2) opgeleverd | 6 augustus 2010 |
| Constructie rapport (iteratie 3) opgeleverd | 27 augustus 2010 |
| Testfase | |
| Testplan opgeleverd | 30 juli 2010 |
| Testplan uitgevoerd opgeleverd | 3 september 2010 |
| Demo opgeleverd | 23 september 2010 |

**Tabel 2: Mijlpalen (visiondocument)**

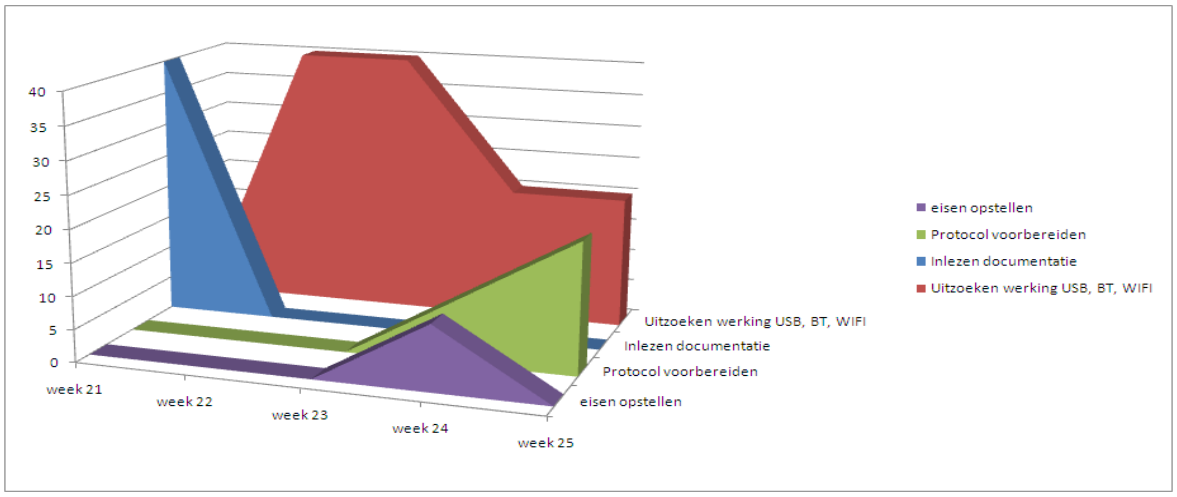
### 3.3.3 Iteratieplan

De planning die ik gemaakt heb bij het schrijven van mijn afstudeerplan is een globale planning. Om wat meer detail in mijn planning te brengen voor de elaboratiefase maak ik ook een Iteratieplan. Doordat mijn elaboratiefase maar uit één iteratie bestaat zal ik één iteratieplan maken

Om het iteratieplan op te zetten heb ik gekeken naar de werkzaamheden die gedaan moeten worden en bepaald welke werkzaamheden er in de volgende fase (elaboratiefase) uitgevoerd moeten worden. Voor de elaboratiefase heb ik vier hoofdwerkzaamheden bepaald.

- Inlezen documentatie;
- Eisen opstellen;
- Protocol voorbereiden;
- Uitzoeken werking USB, Bluetooth, WIFI.

Om structuur aan de elaboratiefase te geven is het niet mogelijk om alle werkzaamheden tegelijk uit te voeren. Daarom heb ik bepaald welke werkzaamheden eerst gedaan moeten worden. De volgorde van werkzaamheden die gedaan moeten worden zijn bepaald aan de hand van de nodige informatie om die uit te voeren. Een voorbeeld is: het is moeilijk om de eisen op te stellen zonder dat er kennis is vergaard over CALM/Spits. Dus dan moet ik me eerst inlezen om vervolgens de eisen te kunnen op stellen. De volgorde van de werkzaamheden die ik ga uitvoeren ziet u in de afbeelding op de hieronder. De getallen op de Y-as van de grafiek zijn het aantal uur die ik van plan ben aan elk activiteit te besteden.




**Figuur 7: Iteratieplan (visiondocument)**

## 3.4 Kwaliteitsplan

Om de kwaliteit van mijn opgeleverde producten en uiteindelijk mijn project te garanderen ben ik van plan om mijn producten die ik oplever te laten controleren door mijn begeleiders. Documenten die specifiek betrekking hebben tot Spits worden aan mijn Spitsbegeleider gegeven ter controle en algemene documenten worden aan mijn Working Tomorrow begeleiders gegeven ter controle.

Na het schrijven van mijn visiondocument heb ik deze conform mijn kwaliteitsplan aan mijn Working Tomorrow begeleider gegeven om te laten lezen en feedback op te geven. Hij had j weinig inhoudelijke feedback. Dit komt doordat ik al bij aanvang van mijn project veel tijd heb gestoken in mijn afstudeerplan.

# 4 Elaboratiefase

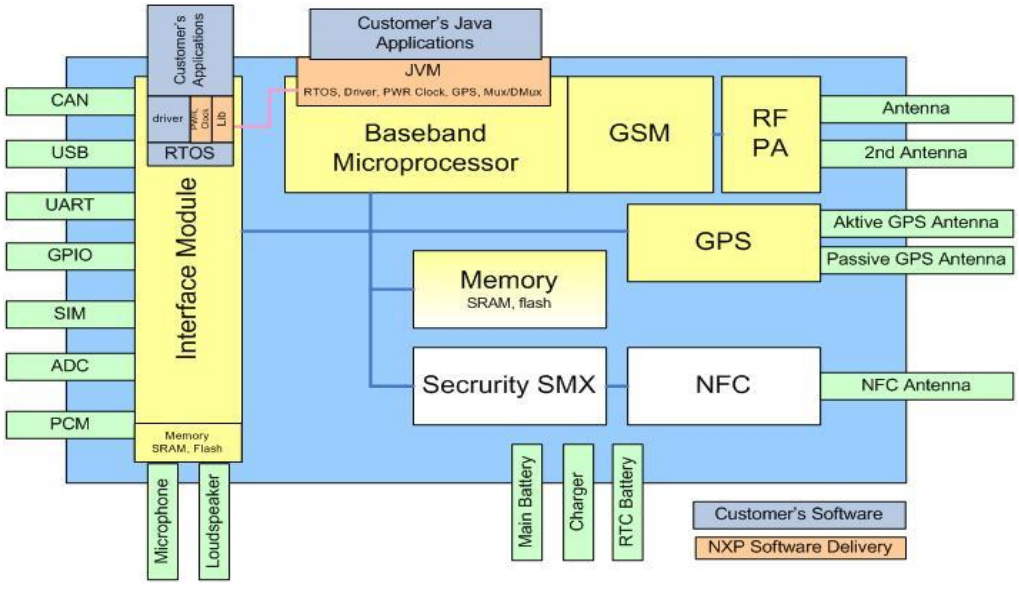
In de literatuur over RUP wordt beschreven wat het doel en de werkzaamheden in de elaboratiefase zijn. [5] In de elaboratiefase wordt het probleem domein bestudeerd, wordt een architectuur gemaakt en worden de grootste risico's beperkt. Aan het eind van de elaboratiefase is een uitgebreide kennis van het gehele systeem vereist. In de elaboratiefase begint het project zijn vorm te krijgen en wordt de mogelijke oplossingen uitgewerkt.

Ontwerpbeslissingen kunnen pas gemaakt worden wanneer men alles weet over het systeem. Met name de scope en de belangrijkste functionele en non-functionele eisen. Aan het eind van de elaboratiefase wordt bepaald of het project door gaat naar de constructiefase en dan de transitiefase.

In mijn elaboratiefase ga ik de werking van de ATOP onderzoeken. Ik wil weten welke communicatiekanalen de ATOP heeft en de programmeertaal die gebruikt moet worden. Na het onderzoeken van de ATOP ga ik USB, Bluetooth en WIFI bestuderen om achter te komen hoe deze communicatiekanalen achter de schermen werken. Ik ga de scope van mijn project verder definiëren, alle use-cases in kaart brengen en beschrijven, aan de hand van gesprekken met mijn Spits begeleider de eisen van de communicatielaag bepalen, verder ga ik de risico's uit de inceptiefase evalueren en bepalen of deze zijn veranderd of dat er risico's zijn bijgekomen. Ik geef ook een voorbeeldoplossing waar de mogelijke oplossing uit de inceptiefase verder wordt uitgewerkt. Aan het eind van de elaboratiefase zal ik de planning nog eens onder de loep nemen en deze eventueel aanpassen waar dit nodig is.

## 4.1 ATOP

Bij het onderzoeken naar de werking van de ATOP ben ik met een consultant van Logica gaan zitten die intensief met de ATOP werkt. Wij hebben een vergadering gepland en in deze vergadering werd mij uitgelegd wat de ATOP allemaal kan. In dit gesprek werd bekend gemaakt dat de ATOP geen Bluetooth kanaal en geen WIFI kanaal heeft. In de afbeelding hieronder kunt u de communicatiekanalen van de ATOP zien.




**Figuur 8: ATOP**

Zoals u kunt zien heeft de ATOP geen Bluetooth en geen WIFI communicatie kanalen. het communicatiekanaal kan alleen worden gemaakt met een kanaal die door beide apparaten wordt ondersteund. Om deze reden ben ik genoodzaakt om de communicatielink alleen via USB te bouwen.

## 4.2 USB

USB staat voor Universal Serial Bus en is ontworpen om poorten zoals de seriële en parallellepoort te vervangen. USB kan randapparatuur aan een PC koppelen. USB is in eerste instantie alleen ontwikkeld voor het gebruik op PC's maar gedurende de jaren wordt USB ook op andere apparaten gebruikt zoals PDA's, smartphones, consoles en zelfs als stroomkabel tussen een apparaat en een AC adapter.

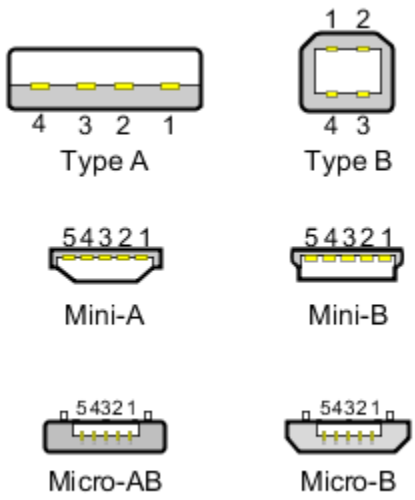
Een USB systeem heeft een asymmetrisch design, bestaande uit een host en een aantal USB-poorten. Doordat USB een groot aantal soorten apparaten ondersteunt wordt er met zogenaamde klas codes gewerkt deze klas codes zijn nodig om de functionaliteit van het apparaat te identificeren en om de geschikte driver te laden voor het apparaat.[7] In de tabel hieronder ziet u een gedeelte van de lijst met de verschillende klas codes met hun beschrijving en een voorbeeld over welk apparaat het gaat. Voor de hele lijst kunt u terecht in bijlage 6.

| Class | Usage | Description | Examples |
|---|---|---|---|
| 00h | Device | Unspecified | (Device class is unspecified. Interface descriptors are used for determining the required drivers.) |
| 01h | Interface | Audio | Speaker, microphone, sound card, MIDI |
| 02h | Both | Communications and CDC Control | Ethernet adapter, modem |
| FFh | Both | Vendor Specific | (This class code indicates that the device needs vendor specific drivers.) |

**Tabel 3: USB class codes [8]**

### USB connectoren

De USB standaard definieert een aantal soorten poorten namelijk USB-A en USB-B. bij de verschijning van USB 2.0 zijn er een aantal poorten bijgekomen MiniUSB-A, MiniUSB-B, MicroUSB-A, MicroUSB-B. in de afbeelding hiernaast kunt u de verschillende poorten zien. Verder zijn er enkele poorten gemaakt die een bedrijfseigen stekker hebben die aan het apparaat gekoppeld moet worden en aan de andere kant een standaard type A stekker voor de PC.




Figuur 9: USB connectoren (wikipedia)

### USB-A

De standaard USB-A stekker is een platte rechthoek die in de USB-host wordt gestoken. Deze stekker zorgt voor stroom en data die voor het apparaat bestemd is. De stekker is vaak te zien aan apparaten zoals toetsenborden en muizen.

#### USB-B

Een standaard USB-B stekker is vierkant met afgeronde hoeken aan de buitenkant. De standaard USB-B stekker wordt gebruikt om apparaten zoals printers aan de host te koppelen. Na de koppeling zal de USB-B stekker stroom en data leveren vanuit de host.

Voor mijn project ga ik gebruikmaken van een USB-mini type B stekker voor de koppeling van het Androidtoestel verder gebruikt de ATOP ook een USB-mini type B stekker voor zijn koppeling. Verdere uitleg over de precieze connectie tussen ATOP en Android wordt in een later hoofdstuk beschreven.

De maximale lengte van een USB kabel is 5 meter. De reden hiervoor is dat de maximale toegestane response vertraging ongeveer 1500ns is. Dit houdt in dat wanneer de instructies van de USB-Host niet binnen deze tijd geantwoord worden zal de USB host deze instructies als verloren beschouwen.[7]. Dit is van belang bij het plaatsten van de ATOP in de auto.

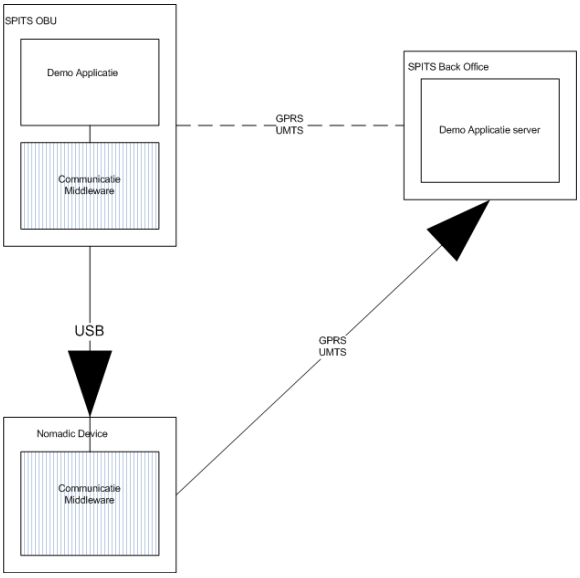
## 4.3 Bluetooth & WIFI

Er is geen mogelijkheid om een USB-Bluetooth adapter te gebruiken op de ATOP. De ATOP en de USB-Bluetooth adapter werken beide in USB-device modus en zoals in hoofdstuk 4.2 beschreven wordt moet er bij een USB connectie altijd een host en een device aanwezig zijn. Doordat het niet mogelijk is om de communicatielink voor de kanalen Bluetooth en WIFI te implementeren heb ik deze kanalen niet meer onderzocht.

## 4.4 Scope

In de inceptiefase heb ik het platform waar ik op ga ontwikkelen vastgelegd, de software die nodig is om te programmeren en de taal waarin ik ga programmeren. Het communicatiekanaal dat gebruikt gaat worden voor de communicatielink moet nog bepaald worden, deze beslissing zal in dit hoofdstuk worden toegelicht.

Na mijn domein onderzoek en het inlezen van de Spits documentatie ga ik een communicatielink bouwen tussen de OBU en een Android smartphone via het USB kanaal. Verder zal ik ook een testapplicatie schrijven. Deze applicatie zal geen diepgaande functionaliteit bevatten. De applicatie zal alleen tekst versturen die door mijn communicatielink naar de Androidtoestel wordt verstuurd en dan via de Androidtoestel via het internet naar de BackOffice. Doordat er ook nog een werkende BackOffice is zal ik ook een BackOffice bouwen die de data zal ontvangen en op het scherm tonen. De data zal niet geïnterpreteerd of verwerkt worden. in figuur 10 ziet u een overzicht van de communicatieprocedure




**Figuur 10: uitleg communicatie procedure**

Zoals u in hoofdstuk 4.2 hebt kunnen lezen heeft elke USB connectie een USB-host kant en een USB-device kant. Het Androidtoestel en de ATOP zijn beide USB-devices. En kan er geen USB connectie opgezet worden. Om dit op te lossen zijn er twee oplossingen mogelijk.

**Oplossing 1:** De USB poort op de ATOP wordt aangepast zodat deze USB-host wordt. Dit houdt in dat de OBU terug naar de fabrikant (NXP) moet gaan zodat deze aangepast kan worden. Er moet dan een nieuwe driver geschreven worden voor de USB poort van de ATOP.

**Oplossing 2:** In de broncode van de kernel wordt USB-host aangezet zodat de Android als USB-host kan worden ingezet. Hiervoor moet de kernel opnieuw gecompileerd worden.

Theoretisch gezien is de keuze om de ATOP de USB-host te maken de betere keuze. Omdat bij het veranderen van de kernel op de Android de garantie vervalt. Verder heeft niet elke nomadic device de mogelijkheid om USB-host te worden.

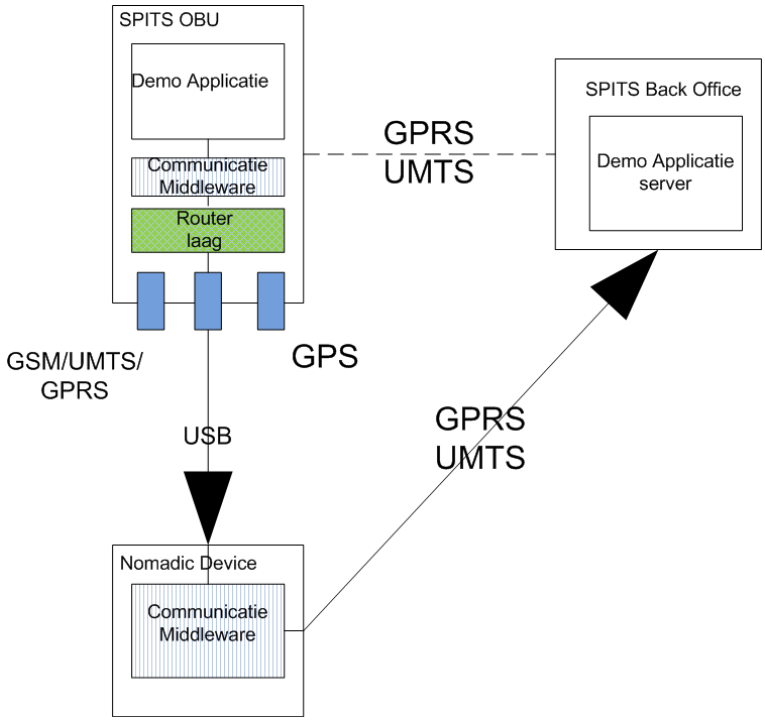
Doordat de Android open source is, is het mogelijk om de broncode in te zien en aan te passen. Dit is niet mogelijk voor de andere nomadic devices. Deze communicatielink is bestemd voor de consument. Het is dan niet mogelijk om de consument te vragen om een ander besturingssysteem te installeren en dat hij zijn garantie op zijn toestel verliest.

Echter gezien het werk dat verricht moet worden om deze keuze werkend te krijgen heb ik samen met diverse experts binnen Logica en mijn mentoren van Logica besloten om toch te kiezen om de Android USB-host te maken. De beslissing is genomen gebaseerd op het feit dat de OBU die door NXP geleverd wordt ook door NXP aangepast moet worden. Deze aanpassing gaat tijd kosten en dit vergroot de kans dat ik mijn project niet op tijd kan afronden.

## 4.5 Architectuur document

In het architectuurdocument breng ik eerst de huidige situatie in kaart. Om dit te doen ga ik naar elk communicatielink van het Spits platform kijken. Ik kijk uitgebreid naar hoe de communicatie tussen de OBU en BackOffice opgezet gaat worden. Doordat er nog geen BackOffice is en nog geen werkende applicaties die een BackOffice gebruiken ga ik alleen kijken wat de protocol tussen OBU en BackOffice inhoudt.

Bij het uitzoeken van het te gebruiken protocol tussen OBU en de BackOffice is naar voren gekomen dat er nog geen uniform protocol bestaat. Er is voorgesteld om de communicatie tussen OBU en BackOffice via een zogenaamde routerlaag te verzenden. Deze routerlaag moet ervoor zorgen dat elk data pakket via het correcte kanaal wordt verstuurd. Het idee is dat de applicatie tegen de routerlaag zegt “Stuur deze informatie naar service X”. De routerlaag moet dan beslissen aan de hand van eerder opgezette regels en/of tabellen via welk communicatiekanaal op de OBU de data verzonden moet worden. in de afbeelding hieronder is dit schematisch weergegeven.




**Figuur 11: uitleg routerlaag**

Omdat de routerlaag die ik ga bouwen geen deel uitmaakt van mijn communicatielaag kies ik ervoor om geen uitgebreide routerlaag te bouwen. Doordat de andere communicatiekanalan nog niet geïmplementeerd zijn heeft het nog geen zin om een uitgebreide routerlaag te bouwen. De routerlaag die ik ga gebruiken is een simpele versie die het concept weergeeft. het bouwen van deze routerlaag zal geen impact hebben op mijn planning.

### 4.5.1 Eisen

Om de eisen te bepalen zijn mijn Spitsbegeleider en ik samen gekomen om de eisen te bespreken. Ik heb zelf eerst een aantal eisen voorgesteld. Tijdens het gesprek tussen mijn begeleider en ik zijn nog enkele eisen naar voren gekomen die van belang zijn voor de communicatielink. De eisen hebben wij samen opgesteld. Nadat wij beiden tevreden waren over de eisen ben ik met de eisen naar mijn bedrijfsmentor gegaan die als tweede expert de eisen heeft doorgenomen. Hieronder de eisen:

**Functionele eisen**

- Middleware moet de data afkomstig van de applicatie inpakken
- Middleware moet de ingepakte data naar de BackOffice sturen.
- Bij uitval van de internet link (tussen de Nomadic Device en BackOffice) moet er een melding naar de applicatie gestuurd worden.
- Bij uitval van de USB link (tussen ATOP en Nomadic Device) moet er een melding naar de applicatie gestuurd worden.
- Als de connectie tussen nomadic device en BackOffice uitvalt moet deze automatisch opnieuw geinitialiseerd worden.
- Er moeten meerdere applicaties tegelijk data kunnen verzenden. (De connectie moet gedeeld worden).

**Niet functionele eisen**

- De link tussen ATOP en nomadic device moet via USB opgezet worden
- De data moet zowel IPv4 als IPv6 ondersteunen.
- De link moet automatisch opgezet worden zonder input van de gebruiker.
- De applicatie stuurt de data synchroon, er moet altijd een response vanuit de ATOP naar de applicatie.
- De middleware draait voorlopig alleen op een Android toestel met name een HTC Hero met Android versie 1.5

**Citaat 7: eisen**

Deze eisen zijn opgesteld toen nog niet bekend was dat de ATOP geen Bluetooth ondersteuning biedt en het was nog niet duidelijk hoe de ATOP de data zou moeten versturen. Toen dit bekend werd is gebleken dat enkele eisen zijn komen te vervallen. Hieronder ziet u de gereviseerde eisen.

**Functionele eisen**

- Middleware moet de data naar de BackOffice sturen.
- Bij uitval van de internet link (tussen de Nomadic Device en BackOffice) moet er een melding naar de applicatie gestuurd worden.
- Als de connectie tussen nomadic device en BackOffice uitvalt moet deze automatisch opnieuw geinitialiseerd worden.

**Niet functionele eisen**

- De link tussen ATOP en nomadic device moet via USB opgezet worden
- De data moet zowel IPv4 als IPv6 ondersteunen.
- De link moet automatisch opgezet worden zonder input van de gebruiker.
- De applicatie stuurt de data synchroon, er moet altijd een response vanuit de ATOP naar de applicatie.
- De middleware draait voorlopig alleen op een Android toestel met name een HTC Hero met Android versie 1.5

**Citaat 8: eisen (Architectuur document)**

### 4.5.2 Use-cases

Use cases opstellen is een van de primaire werkzaamheden van de elaboratiefase. De meeste use cases worden in deze fase bovenwater gehaald, maar er bestaat altijd de kans wanneer je verder werkt aan je project dat er meer use cases boven water komen.[9]. Na het opstellen van de eisen voor de communicatielaag heb ik een aantal use-cases gemaakt. Deze heb ik voor het overzicht in een use-case diagram gestopt. In de volgende afbeelding staat de use-case diagram weergegeven zoals die in het architectuur document staat. Deze use-case diagram met use-case beschrijvingen kunt u in bijlage 7 terugvinden. Bij het maken van use-cases is het belangrijk om te bepalen welke acties use-cases zijn en welke acties juist niet, en het bepalen wie en/of wat de actoren van de use-cases zijn.

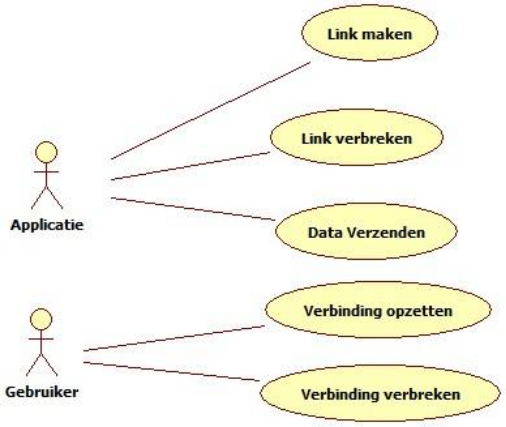
Bij het bedenken van de use-cases was de uitdaging om de alle actoren te identificeren. In de communicatielaag die ik ga bouwen is er interactie tussen verschillende systemen, en interactie tussen systeem en eindgebruiker. Op advies van mijn bedrijfsbegeleider heb ik het boek Writing effective Use Cases gebruikt.

In het boek writing effective Use-cases [10] wordt uitgelegd hoe een actor kan worden geïdentificeerd. Uit de literatuur komt naar voren dat een actor een doel heeft en het systeem heeft de verantwoordelijkheid om het doel van de actor te vervullen.

Bij het bepalen van de actoren zijn er vier kandidaten die een actor kunnen worden het gaat om de volgende kandidaten:

- De applicatie die data wil verzenden
- De eindgebruiker
- De OBU
- De nomadic device

De applicatie heeft 3 doelen om te halen. De doelen die de applicatie heeft zijn een link maken, data verzenden en link verbreken. De eindgebruiker heeft als doel het opzetten en verbreken van de verbinding. De OBU en nomadic device hebben geen van beide doelen om gerealiseerd te worden. In het use-case diagram hieronder kunt u de use-cases zien die ik gemaakt heb voor de communicatielink.




**Figuur 12: use-case diagram (architectuur document)**

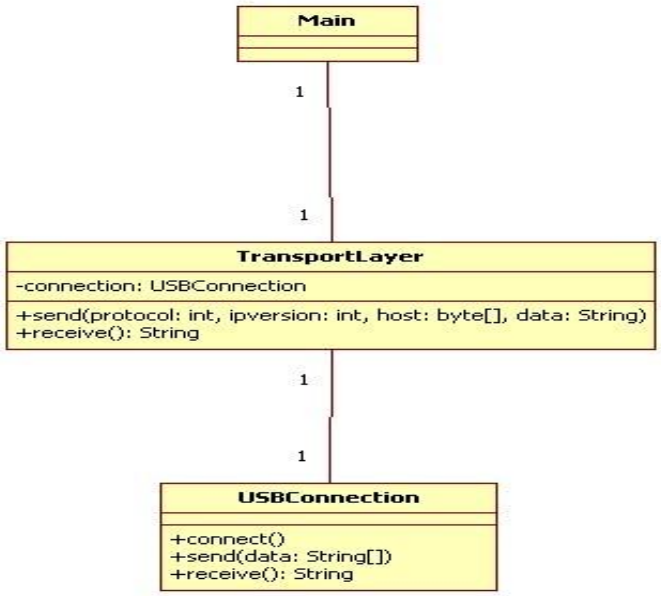
### 4.5.3 Klassendiagram

Klassendiagrammen zijn de ruggengraat van bijna alle object georiënteerde methodes. Dit betekent dat je een klassendiagram bijna altijd kunt gebruiken. [9].

Om structuur aan mijn communicatielink te geven is een klassendiagram noodzakelijk. In de onderstaande figuren ziet u de globale klassendiagrammen die voor de verschillende delen van mijn communicatielink gelden. Doordat ik een communicatielink bouw op twee verschillende platformen heb ik besloten om voor elk platform een eigen klassendiagram te maken. De klassendiagrammen zijn aan het eind van de elaboratiefase nog een concept en kunnen in een later stadium veranderen. De klassendiagrammen voor de BackOffice zal in het volgende hoofdstuk getoond worden.

#### ATOP

Om deze klassendiagram te maken heb ik gekeken naar wat er allemaal nodig is bij het versturen van de data. Toen bekend werd dat het niet mogelijk is om Bluetooth en WIFI te gebruiken als communicatiekanaal heb ik aan een consultant binnen logica gevraagd hoe een USB connectie kan worden gemaakt met de ATOP. Ik ben in contact gebracht met een contact persoon van de ATOP fabrikant. In een E-mail werd uitgelegd dat de ATOP een seriële interface implementeert waar dan gewoon in ASCII kan worden gecommuniceerd. In bijlage 8 staat de email met de instructies en benodigdheden van de fabrikant om een USB connectie te maken. Om Java code op de ATOP uit te voeren moet dit in JavaME geschreven worden. Verder moet de applicatie die op de ATOP draait altijd de naam “main.jar” hebben. Als dit niet het geval is zal de ATOP de applicatie niet uitvoeren. Doordat de ATOP nog een prototype is, is het nog niet mogelijk om meerdere .jar bestanden uit te voeren. Om deze reden is het nog niet mogelijk om onderscheid tussen “routerlaag” en “applicatie” te maken. Ik heb ervoor gekozen om de klassen die ervoor zorgen dat er data transmissie plaatsvindt op te nemen in de testapplicatie. Ik heb hiervoor gekozen omdat ik een proof of concept ga opleveren. Het klassendiagram voor de testapplicatie met routerlaag ziet u hierbeneden. Dit ontwerp wordt voor de ATOP gemaakt.




**Figuur 13: Klassendiagram ATOP (architectuur document)**

Om toch nog het idee te geven dat de testapplicatie gebruik maakt van een routerlaag heb ik in het klassendiagram van de ATOP een extra klasse opgenomen die de functie van de routerlaag moet illustreren. Deze klassendiagram is nog niet compleet maar geeft een idee van de functionaliteit van de testapplicatie. Deze klassendiagram heeft nog niet alle eisen verwerkt doordat dit nog een prototype van het uiteindelijke geheel is.

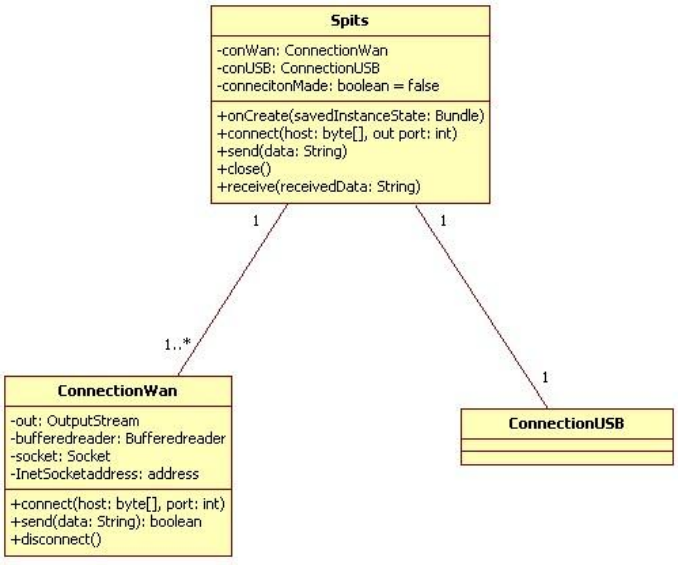
**Main**
De Main klasse is de hoofdklasse van het programma. In deze klasse wordt de data gemaakt en wordt de TransportLayer aangeroepen om de data te verzenden. Andere functionaliteit van deze klasse wat in de constructiefase gemodelleerd en uitgewerkt wordt is het maken en verzenden van het Packet-object.

**TransportLayer**
Deze klasse zou de routerlaag moeten voorstellen. Doordat het nog niet mogelijk is om meerdere applicaties tegelijk op de ATOP te draaien zal deze klasse tijdelijk als routerlaag functioneren De TransportLayerklasse bevat om deze reden dezelfde attributen en methodes als de Main klasse.

**USBConnection**
Deze klasse zal voor de daadwerkelijke USB connectie zorgen. Doordat het nog niet bekend is in deze fase hoe deze connectie opgezet zal worden heb ik 3 klassen bedacht die er zeer waarschijnlijk zeker in moeten. Dit zijn de connect, send en receive methodes.

### Nomadic Device

Bij het bepalen van het klassendiagram voor de nomadic device moet ik rekening houden met twee klassen die connecties moeten afhandelenEen klasse voor de USBconnectie met de naam ConnectionUSB en een klasse voor de internetconnectie. Deze klasse heet ConnectionWAN. Verder is er een derde klasse nodig, de hoofd klasse waar het gehele proces van data ontvangen en data verzenden wordt geregeld. In de afbeelding hieronder ziet u de klassendiagram voor de nomadic device. Dit klassendiagram is terug te vinden in bijlage 9.




**Figuur 14: Klassendiagram nomadic device (architectuur document)**

**Spits**
Bij initialisatie van de applicatie op het Android platform zal eerst de klasse Spits aangeroepen worden. Deze klasse is een subklasse van de klasse *Activity*. De Activity klasse is een speciale klasse binnen het Android platform. Deze klasse is verantwoordelijk voor alle interacties tussen het systeem en de eindgebruiker. Dit betekent dat de Activity klasse de verschillende GUI elementen die de applicatie gebruikt maakt en weergeeft. Er zijn twee methodes die bijna alle subklassen van Activity implementeren en dat zijn onCreate(Bundle) in deze methode wordt je activiteit geïnitialiseerd met de methode setContentView(int) met een lay-out bronbestand waar alle UI elementen in zijn gedefinieerd. Door gebruik te maken van de findViewById(int) kan je de UI elementen programmatisch benaderen. De tweede methode die bijna elke subklasse van Activity implementeert is de onPause() methode, deze wordt gebruikt om alle zaken te regelen die van toepassing zijn wanneer de gebruiker de activiteit verlaat. De Activity klasse is een belangrijk deel van de levensduur van een applicatie. De manier waarop activiteiten worden gestart en samen gebracht is een fundamenteel gedeelte van het applicatie platform.[11]. De Spits klasse zal eerst een connectie met de ATOP via USB maken. Pas wanneer data binnen komt en klaar is voor het verzenden wordt de connectie met de BackOffice gemaakt. Ik heb hiervoor gekozen om het aantal idleconnecties die de BackOffice zou moeten behandelen minimaal te houden.

**ConnectionWan**
De ConnectionWan klasse zorgt voor de communicatie tussen nomadic device en BackOffice. Zoals u kunt zien in het klassendiagram is het mogelijk om meerdere connecties met de BackOffice te maken. Dit voldoet aan de eis dat de connectie over verschillende applicaties verdeeld moet worden. De connectie tussen de ATOP en de nomadic device is een USB connectie dus dit betekent dat er alleen maar een connectie mogelijk is. Daarom is de associatie tussen de klasse Spits en ConnectionUSB een 1-op-1.

De connectie die opgezet wordt met de BackOffice is een TCP connectie. ik heb gekozen voor TCP omdat TCP garandeert dat je data altijd aankomt. Nadat het bericht op de BackOffice is ontvangen zal de BackOffice een bericht terug sturen. Dit bericht is noodzakelijk om door te sturen naar de ATOP, De ATOP blijft namelijk wachten op antwoord van de BackOffice.

**ConnectionUSB**
De connectionUSb klasse wordt in de constructiefase beschreven, doordat het nog niet bekend is hoe ik de USBconnectie tussen de ATOP en de nomadic device moet opzetten. Nog een reden waarom ik de ConnectionUSB klasse pas in de constructiefase kan beschrijven is omdat de ATOP en nomadic device nog niet in staat zijn om een USB connectie op te zetten. In hoofdstuk 4.4 is uitgelegd waarom dit niet mogelijk is. In de volgende paragraaf wordt uitgelegd hoe ik dit probleem ga oplossen.

Voor een duidelijkere versie van het klassendiagram inclusief de klasse beschrijving kunt u terecht in bijlage 9.

### 4.5.4 Oplossing
Zoals ik in hoofdstuk 4.4 heb beschreven ga ik de communicatielink bouwen met het Androidtoestel als USB-host. Voordat ik kan beginnen met het maken van de communicatielink moet ik er eerst voor zorgen dat de Android als USB-host functioneert. Op de Google code website van Android [12] is een aanvraag gestuurd naar Google om deze functie te bouwen voor het Android platform. Op de

website wordt verteld dat de huidige smartphones ongeveer dezelfde hardware specificaties hebben als de computers van 10 jaar terug. En dat in die tijd het ook al mogelijk was om USB apparaten aan de computers te koppelen.

Dit was voornamelijk mogelijk omdat het besturingssysteem hier ondersteuning aan gaf. Hiermee wil de auteur illustreren dat het niet kunnen ondersteunen van USB-host niet aan hardware tekortkomingen ligt maar aan software tekortkomingen.

Als deze functie aan het Android platform wordt toegevoegd geeft dit de mogelijkheid om een groot aantal USB apparaten aan het Android platform te koppelen. Apparaten zoals:

- Normale toetsenborden
- Flash drives
- Printers
- Scanners
- Monitoren
- Tablets
- USB-Hubs

Doordat deze wijziging tot op heden(juni 2010) nog niet is doorgevoerd ga ik op zoek naar alternatieven voor het in staat stellen van USB-host op het Android platform. Bij het uitzoeken van de mogelijkheden ben ik terecht gekomen bij een artikel geschreven door Arnoud Wokke op de website tweakers.net met de titel “Android-smartphone kan na tweak als usb-host dienst doen”. Na het lezen van dit artikel bleek het om de smartphone met de naam “Driod” te gaan van de Amerikaanse fabrikant Motorola. In het artikel wordt beschreven dat het toestel door een tweak USB apparaten kan herkennen en opladen. De functionaliteit is nog beperkt. De tweak werkt via de micro-usb-poort van het toestel. In de volgende kader ziet u een gedeelte van het artikel.

> Gebruikers hebben voor de tweak een autolader, micro-usb-kabel en usb-verlengkabeltje nodig, schrijft de Brit Chris Paget op zijn blog op [13]. De micro-usb-aansluiting van de autolader moet uit elkaar gehaald worden. Vervolgens moet de weerstand eruit gehaald worden, waarna de twee draadjes rechtstreeks worden verbonden. Het resultaat dient als 'dongle' om de host-functie in de Droid te activeren. De Droid kan out-of-the-box als host dienen, maar heeft wel een micro-usb-poort in van het type B, in plaats van het gangbare AB-type voor hosts.
>
> Vervolgens moeten de kabels van de micro-usb-kabel en het usb-verlengkabeltje doorgeknipt worden. Die twee kabels worden verbonden waarna ene einde van de nieuwe kabel in de Droid gestoken kan worden, terwijl usb-apparatuur aan het andere einde ingeplugd kan worden.
>
> De hack werkt vervolgens door de 'dongle' in de Droid te steken tijdens het opstarten. Als het Motorola-logo in beeld komt, moet de dongle er weer uit. Vervolgens kan usb-apparatuur worden aangesloten. Het mounten gebeurt via een terminal-applicatie.
>
> De tweak kent nog veel beperkingen: zo kan de host-functie maar een keer gebruikt worden, waarna de Droid hem automatisch weer uitschakelt. Bovendien zijn er weinig drivers in Android aanwezig voor usb-apparatuur, waardoor usb-apparaten in veel gevallen alleen opgeladen kunnen worden. Ook moet de dongle er precies op het juiste moment worden uitgehaald, omdat anders een bug optreedt.

**Citaat 9: USB-host hack [14]**

Deze oplossing bewijst dat het mogelijk is om op het Android platform een USB-host te maken maar doordat deze oplossing alleen voor de Motorola Droid van toepassing is kan ik deze oplossing niet gebruiken.

Op de website Android-dls.com heb ik weer eens confirmatie gekregen dat het mogelijk is USB-host op het Android platform te implementeren, maar dit brengt wel enige punten met zich mee. Op de website is er een FAQ pagina opgezet.

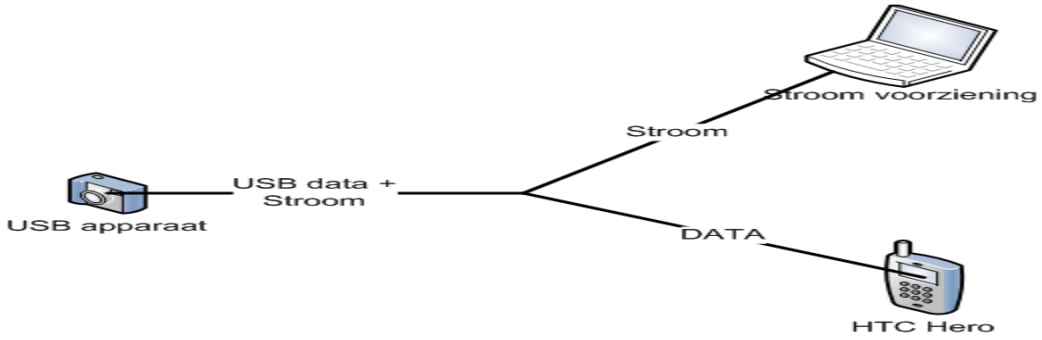
Een van de vragen die op de site staat is precies wat ik ook als vraag heb namelijk: “Has anyone been able to get the G1 to do USB Host mode?” [15]. Als antwoord op de vraag staat het volgende “Short answer: No. Long answer: The chipset supports OTG so it is theoretically possible, but I'm not sure if the G1 hardware would need to be modified. Also, we have no OTG support in the USB driver, and none of the higher level code in android supports host side USB.” [15] Dit wijst erop dat het wel degelijk mogelijk is om USB-host aan te zetten maar dat dit enige aanpassingen in de USB driver van het toestel vergt omdat deze geen USB-host ondersteunt.

Door de zoekmachine Google te gebruiken met de trefwoorden “usb host htc hero” ben ik terecht gekomen op een blog van een programmeur Andrew de Quincey[16]. Op de blog staat dat hij in staat is een HTC hero in USB-host mode te herprogrammeren.

De website legt uit dat de HTC hero USB device ondersteunt alhoewel de auteur het toestel als USB-host wilt gebruiken. Het doel van de auteur is het koppelen van zijn camera aan zijn Android toestel zodat hij betere foto’s kon maken maar nadat hij deze functie heeft aangemaakt op een Android toestel dat het mogelijk is om meerdere USB apparaten aan de Android te koppelen. Zoals USB-seriële apparaten, muizen, toetsenborden enz. Het bleek echter dat het implementeren van USB-host niet zo makkelijk is omdat de auteur geen documentatie had voor de Qualcomm MSM7201A processor.(De processor die in de HTC Hero wordt gebruikt)

Verder moest de auteur ook weten wat voor USB-transontvanger het toestel gebruikt. Uit de code voor de USB-device modus is gebleken dat de HTC hero via het ULPI protocol communiceert. Na het aanpassen van de kernelcode van de HTC hero heeft de auteur een patchfile gemaakt. Deze patchfile moet aangebracht worden aan de HTC hero kernel broncode. Deze code is te vinden op de site http://developer.htc.com/. Wanneer de patchfile aangebracht is moet de code opnieuw gecompileerd worden waar “EHCI/MSM7201 USB host support”wordt aangezet en “USB Function” en “USB Gadget support” worden uitgezet.

De volgende uitdaging is het leveren van stroom aan het USB apparaat die aan de HTC Hero wordt gekoppeld. Doordat de HTC Hero geen stroom kan leveren via zijn USB poort moet er een speciale kabel gemaakt worden. Deze kabel wordt ook wel een Y-kabel genoemd doordat deze 1 uitgang heeft voor data transport, 1 uitgang voor stroom en de andere uitgang voor de USB device dat gekoppeld wordt aan de HTC Hero. In afbeelding 15 ziet u uitleg van de werking van een Y-kabel.




**Figuur 15: Y-kabel uitleg**

De uitvoer van deze stappen ging niet volgens planning. Ik had moeilijkheden bij het compileren van de kernel. Normaliter als je op een computer een programma compileert wordt deze voor het architectuur gecompileerd waarop gecompileerd wordt. Doordat de HTC Hero op ARM-architectuur werkt gaat de gecompileerde code niet werken op het toestel. Hiervoor moet de code worden gecrosscompileerd. Crosscompileren betekent dat het compileren van de broncode voor een van te voren opgegeven architectuur wordt gecompileerd. Bij het maken van een cross compiler zijn enkele stappen vereist. Eerst moeten 4 programma's worden gedownload namelijk: binutils, GCC, newlib en GDB.

- Binutils is een verzameling van programmeertools voor het manipuleren van objectcode in verschillende bestandsformaten.[17]
- GCC is een verzameling van compilers voor C, C++ en Fortran.[18]
- Newlib is een standaard bibliotheek implementatie, bestemd voor gebruik op embedded systemen.[19]
- GDB is de standaard debugger voor de GNU software systeem. [20]

Waneer al deze programmatjes samen werken heet dit een Toolchain. De definitie van een toolchain is:

> "*Een toolchain is een verzameling van programmaatjes in de software ontwikkeling die vaak gebruikt wordt in een bepaalde volgorde zodat de output van een programma de input is voor het volgende programma*."[21]

Nadat ik eruit ben gekomen dat ik een crosscompiler moet gebruiken ben ik op zoek gegaan naar instructies om een toolchain/crosscompiler te configureren voor het Android platform.

Ik ben daarna terecht gekomen op de website http://honeypod.blogspot.com/ 2007/12/compile-android-kernel-from-source.html. op deze website wordt uitgelegd hoe je de Androidkernel vanuit de broncode kan compileren. Het gehele proces wordt in 10 stappen uitgelegd. Als u de stappenplan wilt doornemen kunt u terecht in bijlage 10. Via de website heb ik ook een vooraf geconfigureerde toolchain gevonden. Dit heeft het werk drastisch verminderd. De toolchain is verkrijgbaar op de website www.codeSourcery.com[22] Na het volgen van deze stappen is het gelukt om de Androidkernel te compileren.

Om de reeds gecompileerde kernel te testen moet ik de kernel image op de telefoon uitrollen. Dit heeft ook enige tijd gekost om voor elkaar te krijgen. Op de website theunlockr.com [23][23][23] staat een uitgebreide uitleg van hoe je de kernel image op een Android toestel kan uitrollen. De

belangrijkste stappen die van pas zijn, is het downloaden van een tool met de naam “The Android ROM kitchen” en is te downloaden op de XDA-developers site[24]. Voor de uitgebreide stap voor stap instructies kunt u terecht in bijlage 10.

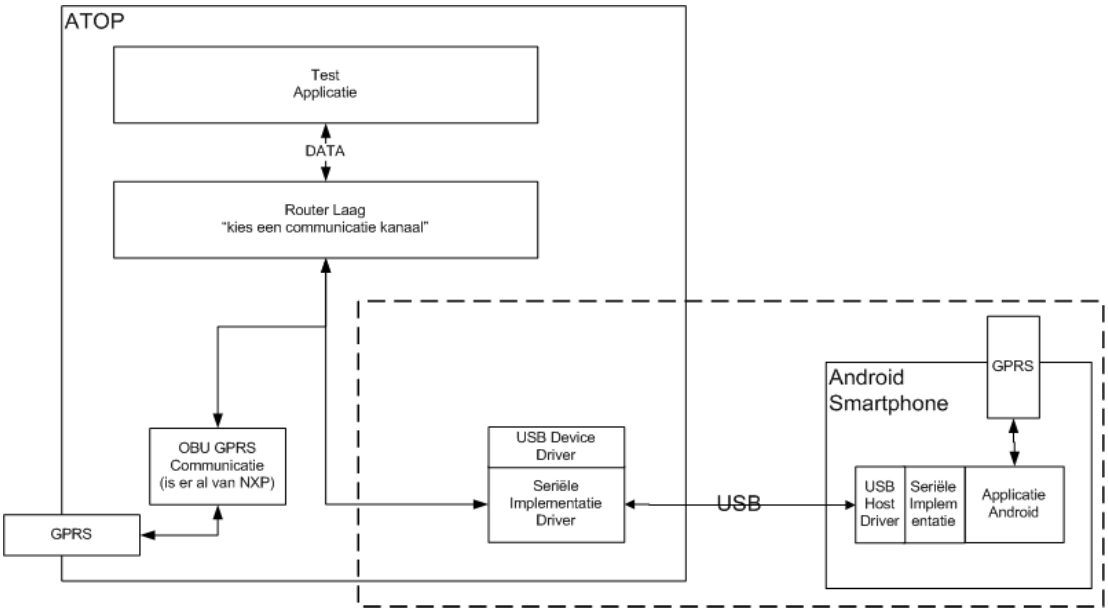
Toen het eindelijk mogelijk was om mijn eigen gecompileerde kernel uit te rollen op het toestel kwam ik erachter dat het werk niet helemaal klaar was. Doordat het niet mogelijk is om een apparaat als USB-host en USB-device tegelijkertijd te laten optreden heeft de programmeur die in eerste instantie de USB-host driver geschreven het mogelijk gemaakt dat de driver handmatig in en uit geladen kan worden. In Linux termen heet dit een module. Bij het uitzoeken van hoe een module in- en uitgeladen kan worden ben ik terecht gekomen op een website die uitlegt dat je de commando "Insmod <modulenaam>" moet gebruiken. Hiervoor is echter wel een shell nodig met superuser rechten.

Nu het Android toestel in staat is als USB-host op te treden moet er een seriële implementatie worden geladen zodat het Androidtoestel met de ATOP kan communiceren. Deze seriële implantatie is te vergelijken met een driver. Om deze driver te laden ben ik op zoek gegaan in de code of er al een USB serieel implementatie bestaat. Tot mijn verrassing ben ik erachter gekomen dat de code al bestaat maar niet gecompileerd wordt. Ik heb daarom de Android kernel opnieuw gecompileerd met ondersteuning van USB serieel en USB host.

Om USB serieel aan te zetten moet deze net als de USb-host driver als module worden ingeladen. Met behulp van de fabrikant van de ATOP ben ik erachter gekomen dat er enkele parameters meegestuurd moeten worden bij het inladen van de USB-serieel module. De parameters die meegestuurd moeten worden zijn de "vendor ID" en "Product ID" van de ATOP.

Bij het koppelen van de ATOP aan het Android toestel bleek dat er om een nog onverklaarbare reden de parameters niet werden ingeladen. Om alsnog USB-serieel te kunnen laden ben ik in de code gaan van de module USB-serieel gaan kijken. Ik ben erachter gekomen dat in de code de twee parameters een waarde krijgen en bij het inladen worden overschreven. Maar dit gebeurde niet bij het inladen op het Android toestel. Ik heb daarom de vendor en product ID van de ATOP hardcoded in de USB-serieel module gestopt. Bij het koppelen van de ATOP aan de Android worden twee seriële poorten gesimuleerd.

In de afbeelding op de volgende pagina ziet u uitgebreid de oplossing die ik voorstel.




**Figuur 16: uitgebreide oplossings voorstel**

In de afbeelding hierboven ziet u schematisch de ATOP afgebeeld met de testapplicatie en de routerlaag die het communicatiekanaal zou moeten kiezen. In het kader ziet u het gedeelte van de gehele opstelling waar mijn opdracht betrekking op heeft. De USB Host Driver en Seriële Implementatie zorgen voor de USB connectie vanuit de Android smartphone en de Seriële Implementatie driver aan de ATOP kant zorgt ervoor dat het mogelijk is dat de ATOP met de Android kan communiceren.

### 4.5.5 Risico's

Bij het uitvoeren van de elaboratiefase is het ook noodzakelijk dat de risicolijst uit de inceptiefase weer erbij wordt gepakt en gekeken wordt of er risico's zijn bijgekomen[5]. Bij het herzien van de risico's bleek dat er nog een aantal risico's zijn bijgekomen. Hieronder vindt u de tabel met de bijgekomen risico's met hun kans dat deze voorkomen, de impact van elk risico en de maatregel die getroffen dient te worden als de risico zich voordoet.

| Risico | Kans | Gevolg | Maatregel |
|---|---|---|---|
| nomadic device ondersteunt geen USB en/of Bluetooth | Klein | Groot | Er moet dan een onderzoek gedaan worden om te kijken waar het probleem ligt. Er kan dan gekozen worden voor een nieuw toestel. |
| ATOP ondersteunt geen Bluetooth | Groot | Groot | De communicatie link tussen ATOP en nomadic device wordt alleen over USB gemaakt. |
| ATOP ondersteunt geen USB | Klein | Groot | Moet er naar een alternatief communicatie kanaal gezocht worden die de ATOP en de nomadic devices wel ondersteunen. |

**Tabel 4: Risico lijst elaboratiefase**

## 4.6 Planning

Uit paragraaf 4.5.4 is gebleken dat het niet meer mogelijk is de planning die ik in de inceptiefase heb gemaakt te hanteren doordat ik de kernel van het Android toestel moet hercompileren. Het hercompileren van de kernel heeft meer tijd in beslag genomen dan ik in eerste instantie heb gedacht. In bijlage 11 kunt u de nieuwe planning vinden. In deze planning ziet u dat de constructiefase iteratie 1 rechstreeks door de elaboratiefase loopt. Ik heb hiervoor gekozen om alvast de communicatielink tussen de nomadic device en de BackOffice te implementeren. Doordat ik in de elaboratiefase dat gedeelte van de communicatielink al heb ontworpen. De tweede iteratie in de constructie fase begint niet direct nadat de eerste iteratie af is. Dit komt doordat ik nog meer tijd nodig heb om de nomadic device klaar te maken voor USB communicatie tussen de ATOP en nomadic device. In de bijlage kunt u de nieuwe globale planning zien.

### Iteratieplan

Om meer structuur in mijn planning te brengen ga ik voor de constructiefase een iteratieplan maken. Net als de iteratieplan in de inceptiefase ga ik eerst kijken welke mijlpalen er behaald moeten worden om de constructiefase succesvol te kunnen afsluiten. De volgende werkzaamheden moeten worden uitgevoerd in de constructiefase Iteratie 1

- Bouwen van software (nomadic device – BackOffice)
- Testplan schrijven
- Schrijven van het ontwerp rapport

In bijlage 12 vindt u de afbeelding met mijn iteratieplan voor constructiefste 1

# 5 Constructiefase

In het boek The Rational Unified Process An Introduction wordt de constructiefase als volgt beschreven.

Gedurende de constructiefase worden alle overgebleven componenten en applicatiefuncties gebouwd en geïntegreerd tot het eindproduct. Alle functies worden grondig getest. De constructiefase is in een zin een vervaardigingsproces waar nadruk wordt gelegd op het managen van middelen en beheersing van activiteiten om de kosten, planningen en kwaliteit te optimaliseren. In de constructiefase verandert het denkbeeld van ontwikkeling van intellectuele eigendom in de inceptie- en elaboratie-fase naar inzetbare producten in de constructie- en transistie-fase. De mijlpalen van de constructiefase zijn het bouwen en testen van de communicatielink. De belankrijkste activiteiten in de constructiefase zijn:

- Complete ontwikkeling van componenten en getest tegen de eisen met behulp van een testplan
- beoordeling van het product tegen de acceptatie criteria van de visie.

Het resultaat van de constructiefase is een product dat klaar is om overgedragen te worden aan de eindgebruikers. Het minimale waar het product aan moet voldoen is dat het product geïntegreerd is op het geschikte platform, de gebruikshandleidingen en een beschrijving van het huidige versie.

Aan het eind van de constructiefase wordt de derde grote mijlpaal behaald. Hier wordt besloten of de software en de eindgebruikers klaar zijn voor de uitrol van de applicatie zonder het project bloot te stellen aan hoge risico's, deze versie heet de *beta* versie. Om de beslissing te nemen of de huidige versie klaar is voor de uitrol moet er aan de volgende criteria voldaan worden: Is het product stabiel genoeg om uitgerold te worden in de productie omgeving? Zijn alle belanghebbenden klaar voor de transitie? En is het project nog binnen budget?

De transitiefase kan worden uitgesteld als deze criteria niet gehaald zijn. [5]

Mijn constructiefase ga ik opdelen in 2 iteraties. In Iteratie 1 ga ik de link tussen de nomadic device en de BackOffice bouwen, in Iteratie 2 ga ik de link tussen de nomadic device en de ATOP bouwen.

In het boek UML distilled wordt uitgelegd dat de interactie van een systeem met twee diagrammen kan worden gemoduleerd. met behulp van sequentiediagrammen of collaboratiediagrammen. Ik kies ervoor om alleen gebruik te maken van de sequentiediagrammen omdat ik hier het meeste ervaring mee heb[9]Na het ontwikkelen van de software is het tijd om te gaan testen. Voor dit project ben ik van plan te gaan testen met de testmethode Testframe. Ik ga hiervoor een testplan opzetten en testgevallen bepalen. Het boek dat ik ga gebruiken als leidraad is "Testframe een praktische handleiding bij het testen van informatiesystemen." [25]

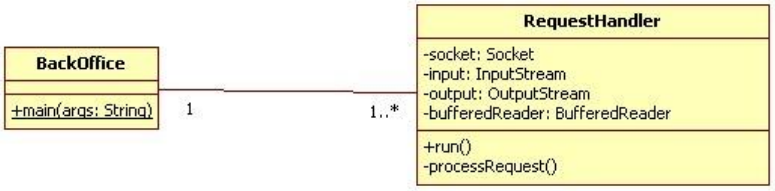
## 5.1 Constructiefase Iteratie 1

### 5.1.1 BackOffice

In hoofdstuk 4.6 is uitgelegd waarom ik genoodzaakt ben van mijn planning af te wijken. Doordat het compileren van de kernel niet voorspoedig loopt heb ik besloten om de eerste iteratie van mijn

constructiefase parallel aan mijn elaboratiefase uit te voeren omdat de nomadic device gewoon gebruik kan maken van de standaard kernel om data te versturen nar de BackOffice. Dit betekent dat de link tussen BackOffice en ATOP ook zonder een zelf gecompileerde kernel werkt.

Doordat er nog geen werkende BackOffice is ga ik ook een BackOffice bouwen. Ik kies ervoor om een simpele BackOffice te bouwen deze BackOffice is een simpele listener die de data dat voor de BackOffice bestemd is ontvangt en op het scherm toont. In figuur 17 ziet u het klassendiagram en in figuur 18 ziet u de sequentiediagram van de BackOffice die ik ga bouwen.




**Figuur 17: Klassendiagram BackOffice**

Het klassendiagram voor de BackOffice heeft twee klassen genaamd BackOffice en RequestHandler. De BackOffice klasse heeft maar een methode, de main methode. In deze methode wordt de server aangemaakt en wordt er gewacht tot wanneer data binnenkomt die bestemd is voor de BackOffice.

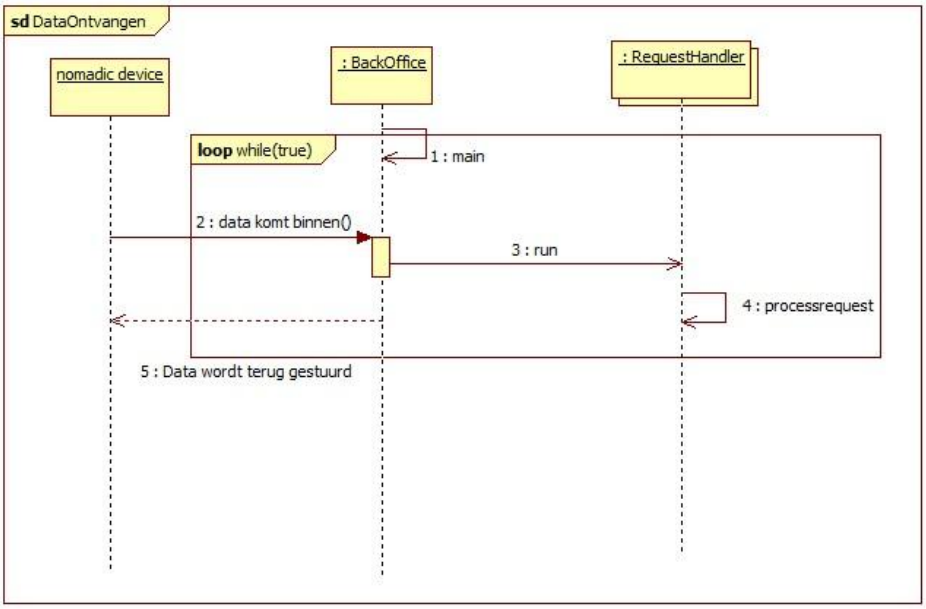
De RequestHandler klasse zorgt ervoor dat binnenkomende data wordt behandeld. De RequestHandler klasse heeft vier attributen: socket van het type Socket, input van het type InputStream, output van het type OutputStream en bufferedReader van het type BufferedReader. Deze attributen maken het mogelijk dat de data ontvangen en uitgelezen kan worden.

De methodes in de Requesthandler zijn run() en processRequest(). De methode run() is een overschreven methode uit de interface Runnable. En de methode procesRequest zal de binnengekomen berichten doornemen en tonen.

Dit klassendiagram heb ik gemaakt door eerst te bedenken wat er nodig is om een basale connectie te maken tussen twee hosts. Hiervoor heb je een Socket om de connectie op te zetten nodig, een InputStream en OutputStream om de data te ontvangen en verzenden. In de eerste versie van de BackOffice zal de BackOffice alleen tekst ontvangen. Om deze tekst efficiënt in te lezen maak ik gebruik van een BufferedReader.

Een van de eisen is dat de internetconnectie van de nomadic device moet worden gedeeld met meer applicaties. Om deze eis te implementeren maak ik een BackOffice die elke nieuwe connectie die gemaakt wordt in een eigen thread stopt. Deze keuze heeft als extra voordeel dat er meerdere systemen tegelijk de BackOffice kunnen benaderen. Om deze reden is de multipliciteit aan de RequestHandler kant in het klassendiagram van de BackOffice 1 of meer.

Om de interactie tussen de verschillende objecten binnen de BackOffice applicatie te moduleren is het maken van een sequentiediagram noodzakelijk. In Figuur 18 ziet u het sequentiediagram van de BackOffice.




**Figuur 18: Sequentiediagram BackOffice**

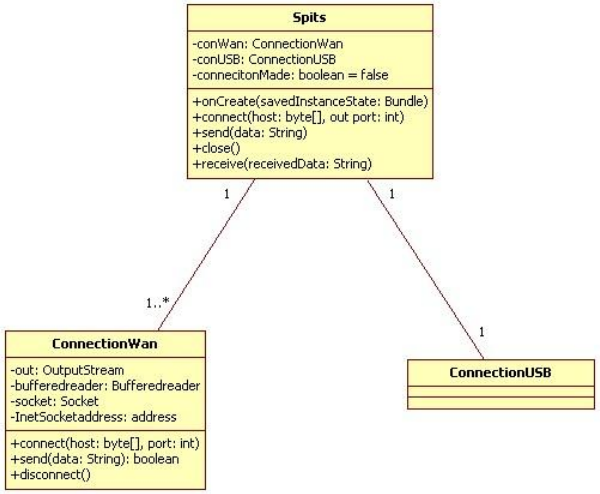
De BackOffice zal bij het opstarten de server initiëren. Wanneer deze geïnitieerd is zal een oneindige lus alle data ontvangen die voor de BackOffice bestemd is. Elke keer wanneer data binnenkomt wordt de RequestHandler aangeroepen en wordt er ook meteen een thread aangemaakt. De thread is nodig om meerder connecties met meerdere applicaties aan te gaan. In de Requesthandler wordt de data die binnen is gekomen uitgelezen en geprint op het scherm.

Doordat de BackOffice een simpele applicatie is die heel weinig eigen functionaliteit heeft heb ik besloten om op het internet op zoek te gaan naar hoe je eenvoudig een listener maakt en omdat ik heel weinig ervaring heb met het programmeren van multi-thread systemen heb ik hiervoor ook een voorbeeld gezocht. De code die ik uiteindelijk gebruikt heb voor de BackOffice kunt u terug vinden op de meegeleverde CD.

Bij het opstarten van de applicatie wordt aan de hand van een ServerSocket een listener aangemaakt. Deze listener luistert op poort 9000. Wanneer de ServerSocket gemaakt is gaat het programma in een oneindige lus. In deze lus blijft de server wachten tot er een connectie met de server wordt gemaakt en dan zal het programma de RequestHandeler aanroepen waar dan de binnengekomen data wordt verwerkt.

### 5.1.2 Android toestel

De code voor het Androidtoestel bestaat uit 3 klassen: De hoofdklasse Spits in deze klasse wordt de Activity voor Android gestart en kan er interactie met de gebruiker plaatsvinden De klasse die voor het transporteren over het mobiele internet kanaal zorgt is ConnectionWan en de klasse ConnectionUSB zal ervoor zorgen dat er data gestuurd kan worden over USB. Deze klasse wordt pas in de volgende iteratie geïmplementeerd en is om deze reden nog leeg.

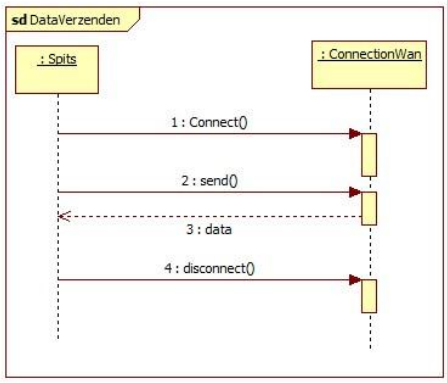



**Figuur 19: Klassendiagram Android (Iteratie 1)**

Dit klassendiagram is voor deze iteratie hetzelfde gebleven als de klassendiagram uit hoofdstuk 4.5.3.

Net als voor de BackOffice ga ik de interactie in het systeem modeleren in een sequentiediagram. Dit sequentiediagram heb ik gemaakt om de interactie bij het verzenden van data naar de BackOffice te tonen. De twee objecten die hiervoor nodig zijn de Spits klasse en de ConnectionWan klasse.

In het sequentiediagram van het Androidtoestel ziet u de interactie tussen de twee klassen die in deze iteratie zijn geïmplementeerd. Bij initialisatie van de applicatie op het Androidtoestel wordt de connectie met de BackOffice gemaakt. Na het maken van de connectie wordt de data verstuurd. en de connectie met de BackOffice weer gesloten. In figuur 20 ziet u het sequentiediagram voor het verzenden van data.




**Figuur 20: Sequentiediagram Nomadic devcie (iteratie 1)**

Bij het schrijven van de code voor het Androidtoestel ben ik te werk gegaan met een Button. Deze button voert de sequentie data verzenden uit wanneer deze ingedrukt wordt. Ik heb hiervoor gekozen omdat ik op die manier makkelijker kan bijhouden welk stukje code wanneer wordt uitgevoerd. Zo heb ik een beter overzicht op de code wanneer er foutmeldingen optreden tijdens het schrijven van de applicatie. Bij afronding van de applicatie in de volgende iteratie zal de button verwijderd worden en zal het opzetten van een connectie, versturen en ontvangen van data en het

verbreken van de connectie automatisch gebeuren. In deze iteratie is het alleen mogelijk om tekst vanaf het Androidtoestel naar de BackOffice te sturen. De broncode van het Androidtoestel is te vinden op de meegeleverde CD.

### 5.1.3 testen

Aan het eind van elke constructiefase moet de opgeleverde applicatie getest worden. Om de tests herhaalbaar en gestructureerd uit te voeren ga ik een testplan opstellen. Het testplan is gemaakt volgens de methode testframe. Ik heb gekozen voor testframe omdat binnen mijn afstudeerbedrijf dit de meest gebruikte testmethode is. Dit geeft als voordeel dat het makkelijk is om een expert te vinden die feedback op mijn testplan kan geven. Ik ga testen om ervan zeker te zijn dat alle eisen en wensen van de opdrachtgever verwerkt zijn in mijn communicatielink.

Ik heb ervoor gekozen om 1 groot testplan te schrijven voor beide constructiefase iteraties. Dit betekent niet dat ik maar een testfase voor de beide iteraties zal hebben.

In mijn testplan heb ik een scope bepaald over de te testen software. De software die ik ga testen is de code die op het Android toestel draait en de Java code die op de ATOP draait. Net zo belangrijk als wat ik wel ga testen heb ik ook beschreven wat ik niet ga testen, omdat het gedeelte wat niet getest wordt niet in de scope van mijn opdracht valt of niet te testen is. De volgende punten zal ik niet gaan testen:

- De software die door de fabrikant van de ATOP wordt geleverd.
- Het inladen van de twee Android kernel modules USB-host en USB-serieel.
- De BackOffice code.

Om het testen zo efficiënt mogelijk te laten verlopen zijn er enkele precondities die gehaald moeten worden voordat er getest kan worden. In het lijstje hieronder ziet u alle precondities die er aan de testfase zijn gesteld.

- Het testen moet op 13 september 2010 beginnen en moet klaar zijn op 17 september 2010.
- De constructie van het systeem is klaar.
- Het testplan moet door alle belanghebbenden worden geaccepteerd
- De testomgeving is op tijd klaar.
- Het testen van de testcases begint niet voordat de precondities zijn gehaald

De tests die ik ga uitvoeren zijn een systeemtest en een acceptatietest. Voor de acceptatietest zal ik de opdrachtgever het gehele product laten testen zodat hij kan beslissen of hij het product die ik ga opleveren accepteert of niet. Deze acceptatietest wordt wel aan de hand van de eisen getest. De specifieke testtechnieken worden later besproken.

Omdat mijn testplan betrekking heeft tot onder andere een acceptatietest heb ik ook enkele acceptatiecriteria opgesteld. Deze criteria zijn samen met mijn Spitsbegeleider opgesteld. Om de acceptatiecriteria meetbaar te maken heb ik bij elk criterium geschreven met welk standaard er gemeten wordt om te bepalen of het criterium bereikt is. In tabel 5 vindt u een overzicht van de acceptatie criteria.

| criterium | standaard |
|---|---|
| De data wordt intact verstuurd en ontvangen door alle onderdelen van het systeem. | Er is voldaan aan de bijhorende eisen. |
| er is foutafhandeling geïmplementeerd | Er is voldaan aan de bijhorende eisen. |

Tabel 5: acceptatiecriteria

Aan het einde van de testfase zullen er een aantal producten moeten worden opgeleverd. De producten die worden opgeleverd zijn de testgevallen, het testrapport met uitkomst van alle testgevallen en de acceptatietest met oordeel van de opdrachtgever. De testgevallen worden in een apart document beschreven en kunt u vinden in bijlage 13.

Bij het opstellen van het testplan zijn de eisen en wensen van de opdrachtgever nodig om te kunnen bepalen wat voor tests er uitgevoerd moeten worden. De documenten die ik doorgenomen heb om mijn testplan te maken zijn mijn visiondocument en architectuurrapport.

Voor de systeemtest ga ik enkele testtechnieken kiezen om de tests uit te kunnen voeren. Ik kies ervoor om de testtechnieken error-guessing en real-life test uit te voeren. Ik kies voor deze technieken omdat deze technieken mijn communicatielink het beste dekken. Testtechnieken zoals de algoritmetest en beslissingstabellen test hebben weinig toegevoegde waarde omdat ik in mijn communicatielink weinig tot geen beslissingensmomenten heb. Mijn broncode is heel erg recht-toe-recht-aan.

De testtechnieken semantische test en syntactische test hebben ook geen toegevoegde waarde omdat er geen datainput is van de eindegebruiker. Mijn applicatie moet zelfstandig draaien zonder invloed van buiten.

**Error-Guessing**
De aanpak van de testtechnieken die ik ga uitvoeren ziet er als volgt uit. Voor error-guessing ga ik bepalen waar de zwakke plekken in het systeem kunnen liggen en daarover testgevallen schrijven. Als zwakke plekken binnen je applicatie kan je denken aan bijvoorbeeld foutafhandeling, illegale waarden die worden opgestuurd en onderdelen van het systeem die meerdere malen onderdeel waren van wijzigingsverzoeken.

Bij het bepalen van mogelijke zwakke plekken in de communicatielink tussen het Android toestel en de BackOffice heb ik geen zwakke plekken gevonden.

**Real-Life test**
In de real-life test wordt getest in een zoveel mogelijk realistische omgeving. Deze test wordt uitgevoerd zoals het product in de productie omgeving zou moeten gaan werken. De concrete testomgeving wordt in de testgevallen beschreven. Werkzaamheden bij het bepalen van de real-life test zijn een profielschets maken en de testgevallen specificeren. het doel van de real-life test is fouten uit de applicatie halen die te maken hebben met het uiteindelijke omvang van het informatiesysteem.

Hieronder een beschrijving van de profielschets van de productie omgeving.

| |
|---|
| De test zal mobiel moeten plaatsvinden doordat de OBU in een auto zal worden geplaatst. Er wordt echter gekozen om de test toch niet in een auto uit te voeren omdat dit geen toegevoegde waarde heeft op de test. De test zal representatief zijn omdat de data vanaf de OBU tot het mobiele apparaat via USB verloopt en dus geen interferentie ondervindt.<br><br>Bij de connectie tussen het mobiele apparaat en de BackOffice zal de sessie niet onderbroken worden bij het veranderen van zendmast. De IP-adressen zijn namelijk statische adressen. De implementatie van het mobiele internet is echter per provider verschillend en is dus provider afhankelijk of het IP dynamisch of statisch is.<br><br>De tests worden uitgevoerd bij T-Mobile als provider. Na contact op te nemen met de technische dienst van T-Mobile is gebleken dat T-Mobile gebruik maakt van statische IP-adressen. Dit betekent dat sessies gewoon open blijven.<br><br>Als omgeving wordt daarom gekozen om de tests in de ontwikkel omgeving te doen. Een andere reden voor deze keuze is de mogelijkheid om de debug board van de OBU te koppelen om zo ook de systeem berichten van de OBU te kunnen lezen. |

**Tabel 6: profielschets (testplan)**

Zoals u in tabel 6 kunt lezen is het niet nodig om de test mobiel uit te voeren. Dit maakt het uitvoeren van de tests wat makkelijker omdat er nog enkele logistieke aspecten geregeld moeten worden zoals stroom voor de ATOP en eventueel de Android. Er is geen mogelijkheid om de systeemberichten zonder een laptop en debug-bord[6] uit te lezen. Om de systeemberichten van de ATOP te kunnen lezen moet er overigens een speciale applicatie gebruikt worden.

Voor een overzicht van alle testgevallen en hun resultaat kunt u terecht in bijlage 14

Als testomgeving heb ik ervoor gekozen om deze in de ontwikkelomgeving te testenomdat het niet mogelijk is om te zien of de tests succesvol zijn of niet doordat er geen output mogelijkheden zijn op de ATOP.Er kan dus niet gezien worden of het pakket dat verstuurd is echt verstuurd is en of deze goed is aangekomen op de BackOffice.

Aan het eind van de testfase zal er een testrapport worden opgesteld met een lijst van alle bugs die tijdens het testen geconstateerd zijn. Deze lijst wordt meegenomen tijdens de acceptatietest met de opdrachtgever.

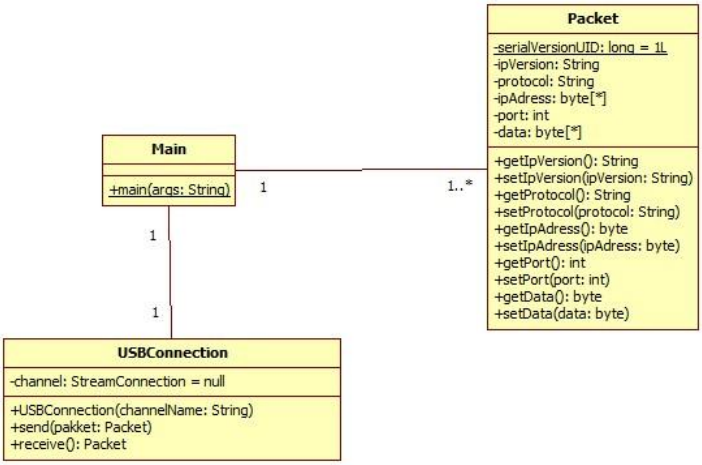
## 5.2 Constructiefase Iteratie 2

Zoals eerder beschreven ga ik in de constructiefase iteratie 2 de USB link bouwen tussen de ATOP en het Android toestel. Ik heb als eerste de software voor de ATOP geschreven omdat dit makkelijker te testen en debuggen is

### 5.2.1 ATOP

Na de vele tegenslagen die ik tijdens het project ben tegengekomen is het eindelijk tijd om de ATOP te programmeren. Voor het programmeren van de ATOP heb ik zoals elke andere software ontwikkelcyclus een klassendiagram en een sequentiediagram gemaakt. De klassendiagram van de ATOP ziet u in figuur 21.

[6] Een debug-bord is een controller die nodig is om je applicatie op de ATOP te krijgen en om eventuele systeemberichten te kunnen uitlezen.




Figuur 21: klassendiagram ATOP Iteratie 2

In dit klassendiagram zijn er 3 klassen gemaakt. Dit klassendiagram heeft twee functionaliteiten namelijk de testapplicatie die de data zou moeten genereren om te kunnen verzenden dit zijn de klassen Packet en Main en het gedeelte van de communicatielink die de ATOP van communicatie moet voorzien dit zijn de klassen USBConnection en Main.

In eerste instantie heb ik het klassendiagram zodanig gemaakt dat de data die verstuurd zou moeten worden een regel tekst zou zijn en in de Main klasse wordt gemaakt. Ik merkte op dat het versturen van tekst niet optimaal zou zijn doordat ik dan elk binnenkomende tekst moet gaan splitsen op speciale tekens. In eerste instantie heb ik bedacht dat de data dat verstuurd moet worden als volgt uit gaat zien:

```
String protocol = "protocol:TCP";
```

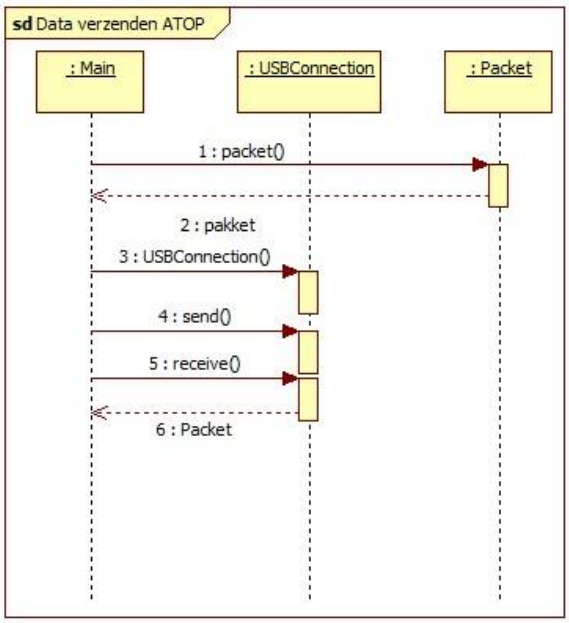
Bij binnenkomst wordt de data gesplitst bij de dubbelepunt. Verwerkingen op tekst zijn zeer inefficiënt. Ik ben toen opzoek gegaan naar alternatieven en heb op advies van een expert serializable objecten gebruikt. Om deze reden is de klasse Packet opgenomen in het klassendiagram.

Serialisatie is het proces waar een datastructuur of een object wordt omgezet in een sequentie van bytes zodat deze kan worden opgeslagen in een bestand, geheugen buffer of verzonden via een netwerk om later weer te worden omgezet naar een volwaardig object. Het tegenovergestelde van serialisatie is deserialisatie, dit betekent de een stroom van bytes om gezet wordt in het oorspronkelijke object.[26] Er zijn wel enkele zaken waarmee er rekening gehouden moet worden bij het gebruiken van serialisatie namelijk het object die geserialiseerd wordt moet ook kunnen bestaan als deze wordt gedeserialiseerd. Dit betekent als bijvoorbeeld een geserialiseerd object verstuurd wordt over het netwerk dat er aan de ontvangstkant ook de mogelijkheid bestaat om een identiek object te kunnen maken.

Het gevolg van de keuze om serialisatie te gebruiken is dat de code van de BackOffice en de code van het Androidtoestel aangepast moet worden. De aanpassingen die gemaakt moeten worden zijn de manier waarop data wordt verzonden en ontvangen. Het klassendiagram van zowel het Androidtoestel als de BackOffice ziet u in hoofdstukken 5.2.2 en 5.2.3.

Net als bij het testen van de connectie tussen het Android toestel en de BackOffice kunt u de testgevallen vinden in bijlage 13

Uiteraard hoort bij elk klassendiagram ook een sequentiediagram die de interactie van het systeem weergeeft. In figuur 22 kunt u de sequentiediagram voor de ATOP zien.




Figuur 22: sequentiediagram ATOP

In het sequentiediagram kunt u zien wat voor interactie er tussen de verschillende objecten binnen de ATOP zijn. Om deze sequentiediagram te maken heb ik een inventaris gemaakt van de stappen die er genomen moeten worden om het systeem in staat te stellen om data te verzenden. De stappen die genomen moeten worden zijn:

1. Het object maken
2. Een connectie opzetten
3. Het object versturen
4. Wachten op antwoord

De code voor het versturen van data heb ik aan de hand van een voorbeeld geschreven die ik van de fabrikant van de ATOP heb gekregen. In het voorbeeld dat ik gebruikt heb werd de data als een stukje tekst verstuurd. Ik heb daarom zelf de code aangepast dat de ATOP geserialiseerde objecten kan versturen. De data die vanuit de ATOP wordt verzonden wordt via een MUX[7] kanaal verzonden. De werking van een MUX kanaal valt buiten de scope van mijn opdracht en kan om deze reden gezien worden als elk andere input- en output-stream die normaliter gebruikt wordt bij het versturen en ontvangen van data. Bij het testen van de applicatie is gebleken dat de data niet altijd goed verstuurd wordt. Na dit na te vragen bij de fabrikant is gebleken dat buffers van het MUX kanaal waarschijnlijk overladen worden met data die verzonden moet worden. Om deze reden heb ik besloten samen met een contactpersoon van de ATOP fabrikant om tijdens het serialiseren van

[7] Multiplexer

het object het object eerst naar een bestand te schrijven en dan het bestand in stukjes van 64 bytes in te lezen en dan te versturen. Dit bleek echter nog steeds teveel data in een keer te zijn. Ik heb daarom besloten om na het versturen van 60 bytes het programma te stoppen voor 100 milliseconden. Deze aanpassing bleek te werken. Dit is echter geen ideale situatie omdat in de productie omgeving vaker gaat voorkomen dat pakketten groter dan mijn testpakket van 252 bytes groot zal zijn. Dit mechanisme zal een beperkende factor op de ATOP zijn. De fabrikant heeft mij wel geïnformeerd dat er een nieuwere versie van de firmware voor de ATOP is die dit probleem zou moeten verhelpen. In verband met tijdsnood is het echter niet gelukt om deze nieuwe firmware uit te proberen. Om de broncode van de ATOP in te zien kunt u deze vinden op de meegeleverde CD.

Om te testen of het programma op de ATOP werkt maak ik gebruik van een Linux computer. Uit de verschillende tests is gebleken dat de ATOP niet altijd dezelfde data verstuurt. De reden hiervoor is nog niet bekend, maar ik ben er redelijk zeker van dat het probleem bij de firmware van de ATOP ligt. De reden waarom ik dit denk is dat het programma wat ik geschreven heb voor de ATOP niet met bytes schuift maar dit is wel het geval in de firmware. De formele tests worden uitgevoerd wanneer het programma op het Androidtoestel klaar is. Een andere reden voor het verdenken van de firmware is het feit dat het programma wel een paar keer foutloos gewerkt heeft.

### 5.2.2 Android toestel

In hoofdstuk 4.5.3 is uitgelegd dat de ATOP bij het koppelen een seriële poort implementeert. Om deze poort aan te spreken vanaf het Androidtoestel zal er een seriële implementatie nodig zijn die het androidtoestel instaat stelt om met de ATOP serieel te communiceren. Ik ben hiervoor op zoek geweest naar een seriële implementatie voor de HTC Hero. Door de zoektermen "usb serial HTC Hero" te gebruiken ben ik terecht gekomen op een project van een programmeur die de HTC hero instaat stelt om seriële data over de USB poort te versturen. Op de pagina wordt beschreven wat er met het project gedaan kan worden. Een aantal HTC Android apparaten hebben een ExtUSB poort. Dit staat voor Extended USB. ExtUSB is een HTC bedrijfseigen stekker dat mini-USB en de mogelijkheid geeft om headsets aan het toestel te koppelen. de ExtUSB poort is verenigbaar met mini-USB en heeft 6 extra pinnen om audio te kunnen ondersteunen[27]. Deze poort wordt ook gebruikt om de telefoon op te laden en te koppelen aan een computer. Naast al deze functionaliteit fungeert de poort ook als een seriële poort waarmee er gedebuged kan worden.

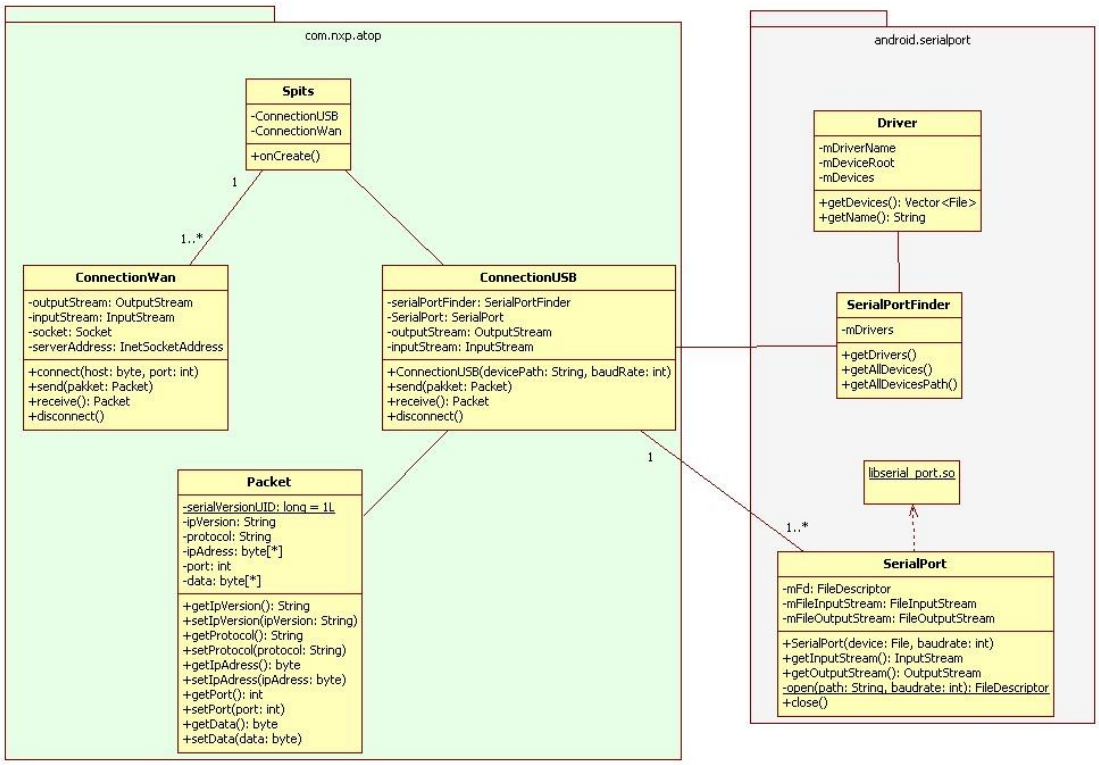
Door USB poort op de HTC hero als serieel fungeert is het mogelijk om de poort te gebruiken als een standaard seriële poort om allerlei seriële apparaten te koppelen. Voordat dit mogelijk is moet de kernelcode voor headset detectie en debug console worden uitgeschakeld. Dit eist wel dat de Androidkernel opnieuw gecompileerd wordt.[28]

Verder heb ik op dezelfde webpagina een demoapplicatie gevonden die de werking van de api bewijst. Het is mij echter niet gelukt om de demoapplicatie werkend te krijgen, maar ik heb de api wel goed kunnen gebruiken. De api bestaat uit twee klassen SerialPort en SerialPortFinder en maakt gebruik van de NDK. De Android NDK staat voor Native Development kit. De Android NDK is een extra bijvulling van de Android SDK. Met deze NDK is het mogelijk om Android applicatieontwikkelaars de mogelijkheid te geven om prestatiekritische delen van hun applicatie in native code te schrijven. De NDK is alleen bedoeld voor gebruik in combinatie met de SDK [29]. De code die de Android NDK gebruikt is een stukje code geschreven in C. Om deze code in te zien kunt deze terugvinden op de meegeleverde CD in de Androidproject in de map jni. De klassen kunt u zien

in het klassendiagram in figuur 23. Nadat het stukje C code geschreven is wordt deze gecompileerd aan de hand van de NDK. Het resultaat na het compileren van de C code is een .so bestand. Een so bestand is bij Linux gebruikers bekend als een gedeelde bibliotheek die door applicaties gebruikt kan worden. Deze bibliotheek moet meegenomen worden tijdens het inpakken van de installatiefile voor een Android apparaat. Door de Android plug-in die geschreven is voor de Eclipse IDE wordt dit automatisch gedaan.

De klassen die ik hierboven genoemd heb SerialPort en SerialPortFinder en de bibliotheek is terug te vinden in het klassendiagram in figuur 23.

Om de USB communicatielink verder te bouwen ga ik nu de klasse ConnecteionUSB in het klassendiagram van het Androidtoestel beschrijven. Verder in het klassendiagram is ook te zien dat de klasse Packet is bijgekomen. Deze klasse implementeert de klasse Serializable zodat de het Packet object verstuurd en ontvangen kan worden.

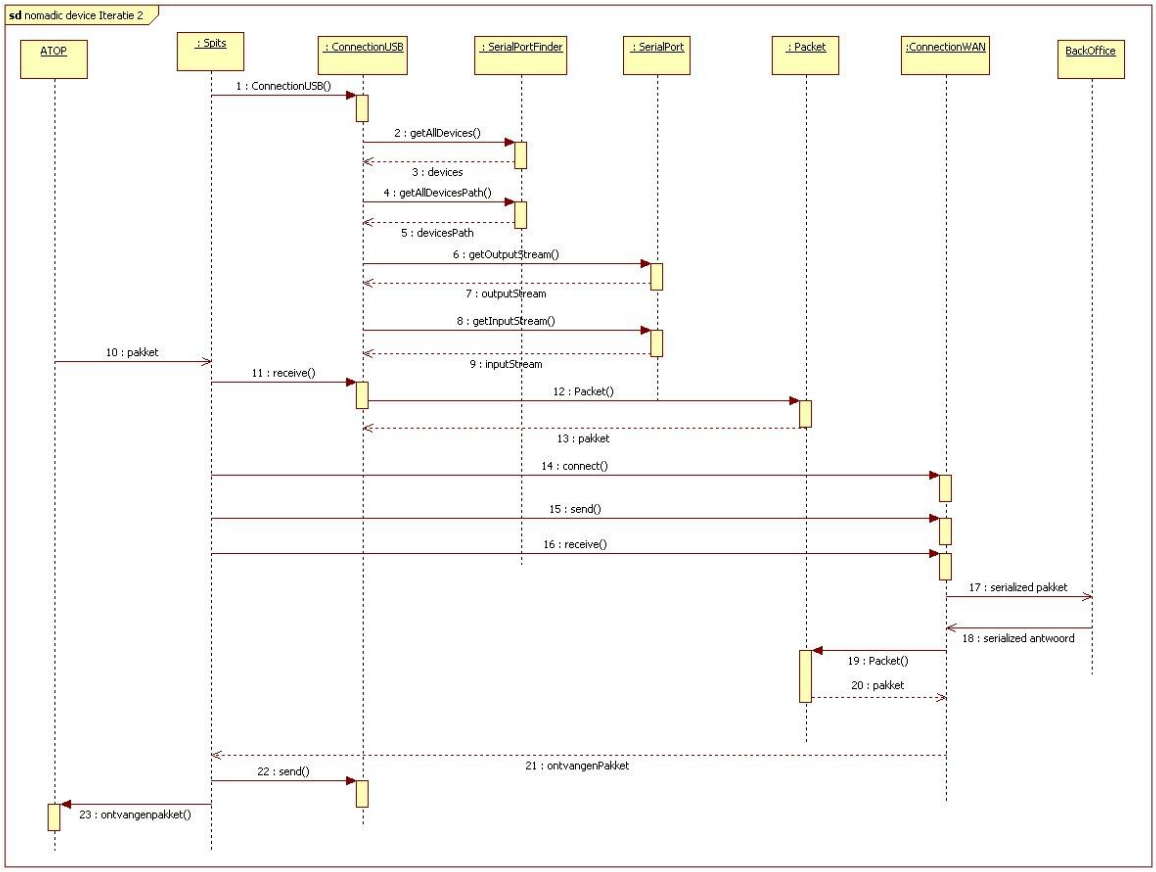



Figuur 23: klassendiagram android toestel Iteratie 2

Dit klassendiagram is ontstaan door het oude klassendiagram opnieuw te gebruiken en de drie klassen die door de API worden geïmplementeerd toe te voegen aan het klassendiagram. De Packet klasse is er ook bijgekomen en ter afsluiting de bibliotheek libserial_port.so.

Bij een dergelijk grote klassendiagram wordt de interactie van het systeem erg interessant ik heb een sequentiediagram gemaakt die deze interactie weergeeft. Ook dit sequentiediagram is opgezet door eerst een inventaris te maken van de stappen die genomen moeten worden om data te ontvangen en verder te sturen. Deze inventaris heb ik gemaakt door te kijken welke stappen er genomen moeten worden om de data te ontvangen en versturen. Bijvoorbeeld niet eerst de

connectie met de BackOffice aangaan wanneer er nog geen data van de ATOP binnen is gekomen. Doordat er aan het klassendiagram is voortgebouwd heb ik besloten om ook op het sequentiediagram voort te bouwen, doordat het einde van het sequentiediagram van het Androidtoestel grotendeels hetzelfde blijft omdat de interactie tussen de verschillende objecten om data te versturen vanaf het Android toestel naar de BackOffice is hetzelfde gebleven. In figuur 24 ziet u het sequentiediagram. Voor een beter overzicht kunt u terecht in bijlage 15.




Figuur 24: Sequentiediagram Nomadic Device Iteratie 2

In dit sequentiediagram wordt de sequentie van het Android toestel weergegeven. Bij initialisatie van de applicatie wordt er eerst een USB connectie gemaakt met de ATOP. Om deze connectie te maken moet er eerst gezocht worden naar het adres dat de ATOP op het Android toestel heeft. Dit wordt gedaan met de getAllDevices methode. Wanneer alle apparaten zijn gevonden worden deze teruggestuurd naar de klasse ConnectionUSB waar dan direct wordt gevraagd voor het pad naar elk apparaat. Wanneer er een apparaat en pad gekozen is wordt de outputstream en de inputstream van het apparaat opengezet zodat er data kan worden verstuurd en ontvangen. Nadat de twee streams zijn aangemaakt wordt er gewacht op het inkomende object vanaf de ATOP. Wanneer deze binnen komt wordt het object ingelezen en worden de BackOffice IP en poort gelezen. Met deze informatie wordt er een connectie opgezet met de BackOffice nadat de connectie is opgezet wordt het object geserialiseerd en verstuurd. Na het versturen van het object gaat de applicatie wachten op antwoord van de BackOffice. Wanneer het antwoord binnenkomt worde het object door verstuurd naar de ATOP.

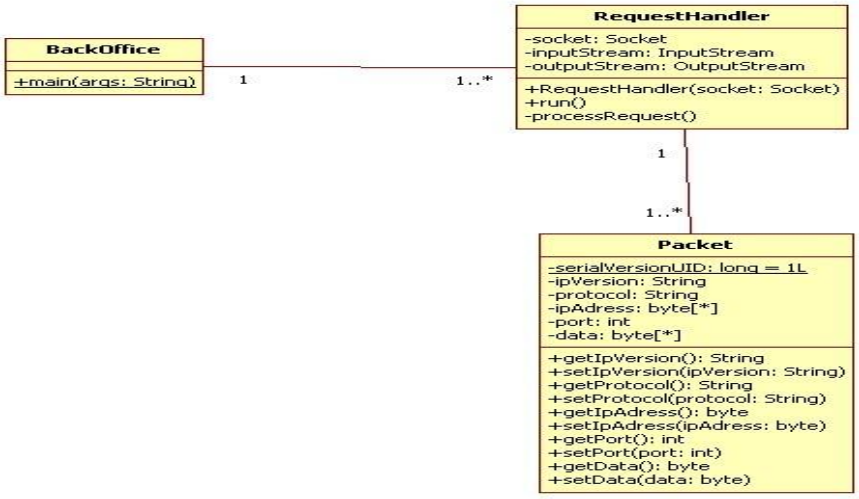
Om de code voor de klasse ConnectionUSB te schrijven heb ik gedeeltes uit de demoapplicatie gebruikt. De gedeeltes van de applicatie die ik gebruikt heb zijn het initialiseren van de inputstreams en outputstreams. Tijdens het programmeren heb ik opgemerkt dat de methodes voor USB communicatie vrijwijl hetzelfde zijn voor zowel de ATOP als het Android toestel. Het enige verschil tussen de methodes is de stream die wordt aangeroepen bij het verzenden of ontvangen van data. Om deze reden heb ik besloten om de code over te nemen.

Om de communicatielink te testen ben ik op zoek gegaan naar mogelijke zwakke punten in de communicatielink. Het bleek dat een mogelijk zwak punt in de link zou kunnen zijn bij het versturen van data via het USB kanaal. Uit de verschillende tests die ik heb uitgevoerd is gebleken dat de dataoverdracht niet altijd correct loopt zoals ik dit in de vorige paragraaf heb beschreven heeft dit waarschijnlijk te maken met de firmware die op de ATOP draait.

De real-life test heeft weinig punten naar voren gebracht die verbeterd moesten worden. Het enige waar nog gekeken moet worden is waarom het versturen van data via de USB link niet altijd goed verloopt.

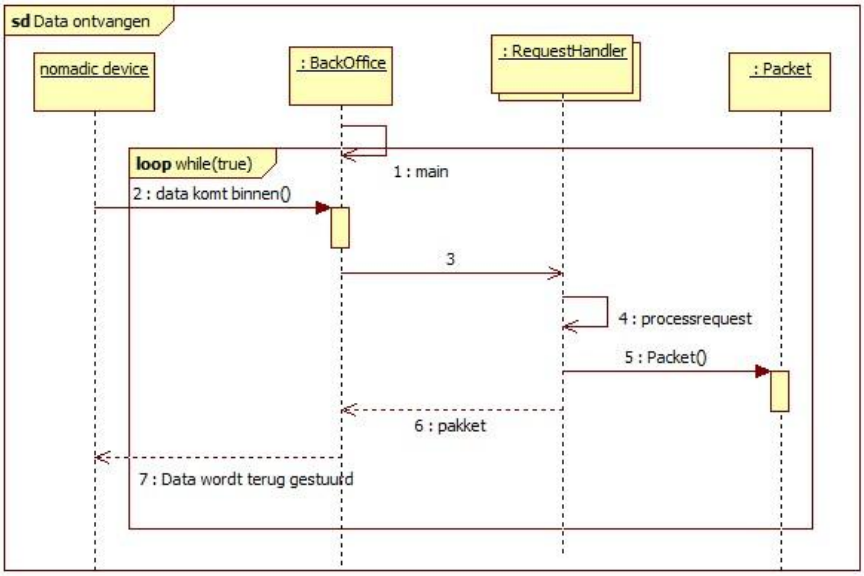
### 5.2.3 BackOffice

Zoals ik eerder heb beschreven ga ik in iteratie 2 van mijn constructiefase de BackOffice een beetje aanpassen. De aanpassing die ik aan de BackOffice ga aanbrengen is: de BackOffice in staat stellen om serialisatie en deserialisatie van het Packet object te kunnen toepassen. Voor de BackOffice is het niet nodig om de geserialiseerde objecten eerst in een bestand te stoppen en dan uit te lezen omdat de BackOffice en het Android toestel beide het object zonder problemen kunnen versturen en ontvangen. In figuur 25 ziet u de herziene klassendiagram voor de BackOffice. In dit klassendiagram ziet u dat de klasse Packet is bijgekomen. Deze klasse is een identieke kopie van de klasse Packet uit het klassendiagram van de ATOP. De multipliciteit die bij de associatie tussen de RequestHandler en Packet hoort is een 1 of 2 associatie. Dit betekent dat er 1 of 2 Packet Objecten per connectie kunnen bestaan. De objecten die kunnen bestaan zijn het inkomende object en het object dat verzonden wordt als antwoord op het object dat binnen is gekomen.




Figuur 25: klassendiagram BackOffice Iteratie 2

De sequentiediagram is ook een beetje veranderd door het toevoegen van de klasse Packet. De verandering heeft geen drastische gevolgen voor de werking van de applicatie. Hieronder ziet u de sequentiediagram zoals die van toepassing is op de BackOffice die aan de Opdrachtgever wordt opgeleverd. De verandering die in dit diagram is bijgekomen is het aanroepen van een nieuw Packet object wanneer het object binnenkomt.




Figuur 26: sequentiediagram BackOffice Iteratie 2

De code van de BackOffice is nagenoeg hetzelfde gebleven als in de vorige iteratie. De verandering die is doorgevoerd is de manier waarop data wordt ontvangen en wat er met de ontvangen data gebeurd.

Doordat ik een gedeelte van de BackOffice heb veranderd zal ik deze weer moeten gaan testen. Voor het voor de tweede keer testen van de BackOffice ben ik op dezelfde manier te werk gegaan door eerst te bepalen waar de zwakke plakken in het systeem liggen. Door de simpliciteit van de applicatie heb ik geen zwakke plekken gevonden. Ik vond dat het programma nu juist robuuster werkt omdat als er om een of andere reden geen correcte data verstuurd wordt zal het systeem niet reageren en de connectie meteen afbreken.

Om de real-life test uit te voeren ga ik wachten tot alle delen van de gehele communicatielink klaar en testbaar zijn.

# 6 Evaluatie en aanbevelingen

In dit hoofdstuk wordt een terugblik geworpen op mijn werkzaamheden gedurende mijn afstudeerproject te Logica. Ik ga eerst enkele zaken onder de aandacht brengen die van belang zijn voor het vervolg van mijn project en tot slot een evaluatie van mijn werkzaamheden met eventuele geleerde lessen.

## 6.1 Evaluatie

Mijn project is net als veel afstudeerprojecten begonnen met een leuk idee en veel goede moed. Tijdens het project kwam naar boven dat niet elk aspect van mijn project even makkelijk en/of mogelijk was. Na veel tegenslagen is het mij gelukt om alsnog een compleet afstudeerproject uit te voeren.

### 6.1.1 Proces

In deze paragraaf ga ik het proces wat ik heb doorlopen weer even kort samenvatten en mijn leermomenten benomen.

Bij aanvang van mijn project tijdens het oriënteren werd het al gauw duidelijk dat ik met een software ontwikkel project bezig zou zijn. hiervoor heb ik dan een geschikte methode moeten kiezen. Ik heb voor de methode RUP gekozen. ik ben blij met de keuze voor RUP omdat deze methode mij de mogelijkheid heeft gegeven om mijn fases grotendeels zelf in te vullen omdat ik aan het begin nog niet wist hoe ik mijn project zou gaan afbakenen. De keuze voor RUP heeft mij ook in staat gesteld om alvast te beginnen met mijn eerste iteratie van mijn constructiefase om zo niet al teveel kostbare tijd te verspillen.

Buiten het feit dat het project procesmatig goed verliep ben ik minder blij hoe het proces inhoudelijk verlopen is. Ik had veel eerder moeten vragen om hulp bij het compileren van de Android kernel. Alhoewel het achteraf bleek dat niemand binnen mijn afstudeerbedrijf een klaar antwoord voor mij had. De reden waarom ik een klaar antwoord had gewild na 6 weken tijd is omdat ik aan het eind van mijn project toch wel een beetje in tijdnood ben gekomen. En dit had voorkomen kunnen worden als ik samen met iemand had kunnen werken die hier meer kennis van heeft.

Ik begrijp wel dat mijn project zeer innovatief is en dat het moeilijk is om iemand te vinden die hier een antwoord op zou kunnen hebben. Mijn leermomentje hier was om toch iemand te zoeken en dan eventueel samen te zitten om tot een oplossing te komen.

Een ander leermoment voor mij was bij de tussentijdse beoordeling toen aan het licht werd gebracht dat ik voor elke constructiefase een testfase moet hebben. Ik had in eerste instantie bedacht om na de twee constructiefases mijn testfase uit te voeren. De reden hiervoor was omdat er aan het eind van de constructiefase en dus testfase bruikbare code is die direct ingezet zou kunnen worden. Maar doordat mijn communicatielink op 3 platformen draait heb ik voor mezelf bedacht dat ik alle delen van de communicatielink bouw en dan pas ga testen. Ik heb alsnog kunnen testen aan het eind van de eerste iteratie omdat ik tijdens het tussentijds beoordelingsgesprek net mijn eerste iteratie aan het afronden.

### 6.1.2 Producten

Ik zal elk document dat ik heb opgeleverd weer één voor één evalueren en kijken wat ik in een volgende keer beter kan uitvoeren en bepalen waar mijn grootste leermomenten zaten.

Tijdens mijn afstudeerperiode heb ik 4 producten opgeleverd: een plan van aanpak/visiondocument, een architectuurrapport/elaboratierapport en een ontwerprapport/constructierapport en heb ik een demo geprogrammeerd.

#### Visiondocument

Bij het schrijven van mijn visiondocument moest ik een risicoanalyse doen. In deze analyse zijn enkele risico's naar voren gekomen. Maar het bleek later dat dit niet alle risico's waren. Het bleek dat er nog wat risico's verscholen waren die pas op een later moment naar boven zijn gekomen. Deze risico's kon ik wel in de elaboratiefase opvangen maar het had beter geweest als ik al vanaf de inceptiefase wist dat er zo een grote risico zou zijn die zoveel tijd zou gaan kosten.

Om de scope te definiëren heb ik ook wat ongebruikelijks gedaan. Ik heb namelijk een deel van mijn scope in de inceptiefase gedaan en een ander gedeelte in de elaboratiefase. Dit heb ik zo gedaan omdat ik in de inceptiefase nog niet wist wat de omvang van mijn project was. Ik moest hiervoor eerst inlezen om zo een idee te krijgen van het project en dan pas in detail de scope definiëren.

#### Architectuurrapport

In het architectuurrapport worden de eisen normaliter beschreven maar het kwam pas later aan het licht dat de OBU geen Bluetooth ondersteunt en om deze redenen moesten de eisen ook aangepast worden. De fout lag aan mijn kant omdat ik niet vanaf het begin ben gaan vragen wat de hardware specificaties van de OBU zijn. Ik heb namelijk aangenomen dat de OBU Bluetooth ondersteunt omdat dit als een mogelijk communicatiekanaal was bij het opzetten van mijn opdracht. Dit sluit helemaal aan bij ook een belangrijke leerpunt: vooral in de beginfase een uitgebreid en diep vooronderzoek doen. Het vooronderzoek wat ik gedaan heb was net niet breed genoeg en elke keer net een beetje te laat bij het vergaren van kennis. Bijvoorbeeld pas nadat ik de eisen heb bepaald erachter komen dat de OBU geen Bluetooth heeft. Pas nadat ik een concept heb voorgesteld erachter komen dat het niet mogelijk is om zomaar USB connectie op te zetten doordat er altijd een Host en Slave nodig is.

#### Ontwerprapport

Het opzetten van mijn ontwerprapporten verliep voorspoedig. Ik had wel enige moeite met het bepalen wat er nou precies in moet en wat er juist niet in moest. Ik heb maar besloten om alleen klassendiagrammen en sequentiediagrammen in de documenten te stoppen.

#### Testplan en testgevallen

Mijn testplan en testgevallen zijn aan de hand van testframe uitgevoerd. Ik vind dat het testplan eerder geschreven moest worden maar doordat ik druk bezig was met het uitzoeken hoe de Androidkernel gecompileerd moest worden en hoe ik de Android in staat kon stellen om met de OBU te communiceren, ben ik even het beeld kwijt geraakt en heb ik niet goed naar mijn tijd gekeken. De les die ik hier geleerd heb is als je project niet verder gaat, je toch nog verder moet gaan met de documenten die voor die fase afgemaakt moeten worden. Het is uiteindelijk wel gelukt om een testplan op te zetten en de testgevallen te bepalen om zo de tests te kunnen uitvoeren.

#### Demo

Ik ben erg blij met het resultaat van mijn demo alleen dat er nog wat puntjes in zitten waar ik geen raad mee weet. Het komt namelijk voor dat de OBU niet elke keer dezelfde data opstuurt. Dit komt door een timingprobleem in de firmware van de OBU. Met de fabrikant heb ik veel contact gehad om te kijken hoe het Androidtoestel de OBU moet herkennen.

## 6.2 Conclusie

In dit hoofdstuk ga ik mijn conclusie van mijn gehele project trekken door mijn probleemstelling naast de doelstelling leggen en kijken of ik deze behaald heb.

Zoals ik in hoofdstuk 3 heb beschreven ziet mijn probleemstelling er als volgt uit:

> *Om onafhankelijk te kunnen zijn van een specifieke Telecom leverancier, willen we de mogelijkheid hebben om ook via een ander kanaal datacommunicatie te realiseren. Zonder gebruik te maken van de communicatie mogelijkheden van de OBU.*

De doelstelling van mijn project ziet er als volgt uit:

> *Het realiseren van een proof of concept van een middlewarelaag om datacommunicatie van de OBU met de back-office tot stand te brengen via een nomadic device*

Als ik deze twee vergelijk met name de doelstelling naar de huidige stand van zaken denk ik wel dat ik deze gehaald heb doordat ik een proof of concept heb gebouwd die inzetbaar is als middlewarelaag. Het is nog niet helemaal mogelijk om meerdere applicaties op de OBU te draaien maar dit is een beperking van de OBU en niet van mijn applicatie. Er zijn nog wel enkele punten die uitgewerkt moeten worden om de OBU echt klaar te maken voor de consumentenmarkt. met name het bufferprobleem waar steeds andere data wordt verstuurd. Een ander punt waar naar gekeken moet worden is het feit dat de nomadic device niet als usb-host mag optreden doordat de consument zijn garantie verliest.

## 6.3 Aanbevelingen

De communicatielink heeft veel potentiële mogelijkheden in de consumentenmarkt. maar voordat de communicatielink levensvatbaar is voor de consumentenmarkt moeten enkele punten worden herzien. In huidige situatie is gekozen voor het Androidtoestel als USB-host, zoals ik in hoofdstuk 4.4 heb beschreven heeft dit enkele nadelen. Om het Androidtoestel instaat te stellen om USB-host te worden moet de kernel worden gecompileerd. Dit brengt met zich mee dat de garantie op het toestel vervalt.

Verder moet er nog onderzoek worden gedaan naar een andere oplossing om de data via USB te versturen. om gebruik te maken van de huidige oplossing zijn er twee dingen nodig op het Androidtoestel: de usbserial module moet geladen worden en er moet een seriële implementatie op het Androidtoestel worden gebruikt zoals beschreven in hoofdstuk 5. Deze seriële implementatie heeft 'superuser' rechten nodig om de seriële poorten op het androidtoestel op te vragen. Maar deze rechten zijn niet beschikbaar wanneer de kernel niet zelf gecompileerd is.

Ik raad om deze redenen aan om toch onderzoek te doen naar de mogelijkheden om de ATOP als USB-host te laten treden.

## 6.4 Competenties

In deze paragraaf worden alle te bewijzen competenties op een rij gezet en gekeken naar hoe deze nou zijn bewezen. Ik hiervoor de STARR methode gebruikt.

### 6.4.1 Analyseren van het probleemdomein

| | |
|---|---|
| Situatie | Er is een vraag naar een additionele communicatielink tussen een OBU en de BackOffice |
| Taak | Ik ga het probleem domein bestuderen en een beslissing nemen naar wat de eind situatie zal zijn. |
| Activiteiten | In de elaboratiefase heb ik uitgebreid uitgezocht hoe de OBU via USB communiceert en deze in een diagram getekend om zo het probleemdomein af te bakenen naar een meer beheersbaar formaat. |
| Resultaat | Goede kennis van de mogelijkheden van de OBU en het gehele Spits platform in zijn algemeen. |
| Reflectie | Het analyseren van het probleemdomein verliep stroef doordat het probleemdomein na elk probleem steeds verdiepte. Maar het is uiteindelijk gelukt om de tegenslagen wel op te lossen. |

### 6.4.2 Realiseren van software

| | |
|---|---|
| Situatie | De eisen zijn bekend duidelijk en de elaboratiefase is afgerond |
| Taak | Ik ga nu aan de constructiefase beginnen om de communicatielink te schrijven. |
| Activiteiten | Aan de hand van eerder gemaakte klassendiagrammen en sequentiediagrammen ga ik in Java de communicatielink bouwen. door de Java API te gebruiken kan ik onbekende methodes doornemen en inzien hoe deze gebruikt moeten worden. Deze activiteiten heb ik in de constructiefase bewezen. |
| Resultaat | Een communicatielink die data via een USB link data verstuurt naar een Android toestel die in zijn beurt weer deze data doorverstuurd naar een zelf gemaakte server. |
| Reflectie | Het bouwen van het programma ging redelijk soepel en ben blij met het geleverde resultaat. |

### 6.4.3 Testen van software systemen

| | |
|---|---|
| Situatie | De communicatielink is klaar om te worden getest |
| Taak | Ik ga testen of de communicatielink aan de eisen van de opdrachtgever voldoen |
| Activiteiten | Bij voorbaat een testplan opstellen waarin wordt verteld wat en hoe er getest gaat worden. Na het opzetten van de testplan heb ik een aantal testgevallen opgezet die ik dan heb uitgevoerd en ingevuld. Deze activiteiten zijn terug te vinden aan het eind van de constructiefase |
| Resultaat | De tests hebben enkele fouten in het programma aan het ligt gebracht maar deze bleken later niet aan het programma te liggen maar aan de firmware wat niet optimaal geprogrammeerd is |
| Reflectie | Ik vond het moeilijk om de testplan op te zetten omdat er zoveel testtechnieken beschikbaar zijn en het kiezen van de geschikte techniek was een punt waar ik veel aandacht aan moest besteden |

# Literatuurlijst


[1] SPITS., What is Spits? *https://spits-project.com.* [Online] 2010. [Citaat van: 10 July 2010.] https://spits-project.com/index.php?option=com_content&view=article&id=66&Itemid=41.

[2] —. On Board Unit. *spits-project.com.* [Online] 2010. [Citaat van: 10 july 2010.] https://spits-project.com/index.php?option=com_content&view=article&id=64&Itemid=56.

[3] NXP., ATOP. *www.nxp.com.* [Online] 2010. [Citaat van: 17 augustus 2010.] http://www.nxp.com/#/pip/pip=[pfp=56424]|pp=[t=pfp,i=56424].

[4] SPITS., Back Office (BO). *spits-project.* [Online] 2010. [Citaat van: 10 July 2010.] https://spits-project.com/index.php?option=com_content&view=article&id=60&Itemid=58.

[5] Kruchten, Philippe., *The Rational Unified Process An Introduction.* s.l. : Addison-Wesley, 2001.

[6] Passchier, Igor., *Spits Network Architecture V0.9.* 2010.

[7] Wikipedia., Universal Serial Bus. *wikipedia.* [Online] 2005. [Citaat van: 26 July 2010.] http://en.wikipedia.org/wiki/Universal_Serial_Bus.

[8] USB.org., USB Class Codes. *www.usb.org.* [Online] 17 november 2009. [Citaat van: 26 augustus 2010.] http://www.usb.org/developers/defined_class.

[9] Scot, Martin Fowler & Kendall., *UML Distilled A Brief Guide To The Standard Object Modeling Language.* sl : Addison-Wesley, 2000.

[10] Cockburn, Alistar., *Writing effective Use Cases.* sl : Addison-Wesley Pearson Education, 2002.

[11] Android Developers., Activity. *Android Developers.* [Online] Google.com. [Citaat van: 17 september 2010.] http://developer.android.com/reference/android/app/Activity.html.

[12] , *Google code android.* [Online] 30 Mei 2008. [Citaat van: 28 mei 2010.] http://code.google.com/p/android/issues/detail?id=738.

[13] Tombom.co.uk., USB Host mode on Motorola Droid. *tombom.co.uk.* [Online] 9 februari 2010. [Citaat van: 7 juni 2010.] http://www.tombom.co.uk/blog/?p=124.

[14] Tweakers.net., Android-smartphone kan na tweak als usb-host dienst doen. *Tweakers.net.* [Online] 10 februari 2010. [Citaat van: 7 juni 2010.] http://tweakers.net/nieuws/65502/android-smartphone-kan-na-tweak-als-usb-host-dienst-doen.html.

[15] android-DLS., Android FAQ. *android-dls.com.* [Online] 9 Juni 2009. [Citaat van: 7 juni 2010.] http://android-dls.com/wiki/index.php?title=Android_FAQ#Q:_Has_anyone_been_able_to_get_the_G1_to_do_USB_Host_mode.3F.

[16] Quincey, Andrew de., Andrew de Quincey's livejournal. *livejournal.* [Online] 9 februari 2010. [Citaat van: 26 juli 2010.] http://adq.livejournal.com/95689.html.

[17] Wikipedia., GNU Binutils. *www.Wikipedia.org.* [Online] [Citaat van: 9 augustus 2010.] http://en.wikipedia.org/wiki/Binutils.

[18] wiktionary., GCC. *www.wiktionary.org.* [Online] [Citaat van: 9 augustus 2010.] http://en.wiktionary.org/wiki/GCC.

[19] wikipedia., Newlib. *wikipedia.org.* [Online] [Citaat van: 9 augustus 2010.] http://en.wikipedia.org/wiki/Newlib.

[20] —. GNU Debugger. *www.wikipedia.org.* [Online] [Citaat van: 9 augustus 2010.] http://en.wikipedia.org/wiki/GDB.

[21] —. Toolchain. *www.wikipedia.org.* [Online] 14 maart 2010. [Citaat van: 9 augustus 2010.] http://en.wikipedia.org/wiki/Tool_chain.

[22] CodeSourcery., Download sourcery G++ lite edition for arm. *codesourcery.* [Online] [Citaat van: 9 augustus 2010.] http://www.codesourcery.com/sgpp/lite/arm/portal/subscription?@template=lite.

[23] The Unlockr., How To: Create Your Own Custom ROM for Android, Part 1 – Setting Up The Android Kitchen. *theunlockr.com.* [Online] 3 maart 2010. [Citaat van: 9 augustus 2010.] http://theunlockr.com/2010/03/26/how-to-create-your-own-custom-rom-for-android-part-1-setting-up-the-kitchen/.

[24] xda-developers., htc android kitchen. *forum.xda-developers.com.* [Online] 15 februari 2010. [Citaat van: 9 augustus 2010.] http://forum.xda-developers.com/showthread.php?t=633246.

[25] CMG., *Testframe een praktische handleiding bij het testen van informatiesystemen.* Den Haag : ten Hagen & Stam Uitgevers, 2001.

[26] MSDN., Serializing Objects. *MSDN.* [Online] Microsoft. [Citaat van: 18 september 2010.] http://msdn.microsoft.com/en-us/library/7ay27kt9(VS.71).aspx.

[27] XDA developers., ExtUSB. *forum.xda-developers.com.* [Online] 21 july 2010. [Citaat van: 19 september 2010.] http://forum.xda-developers.com/wiki/index.php?title=ExtUSB.

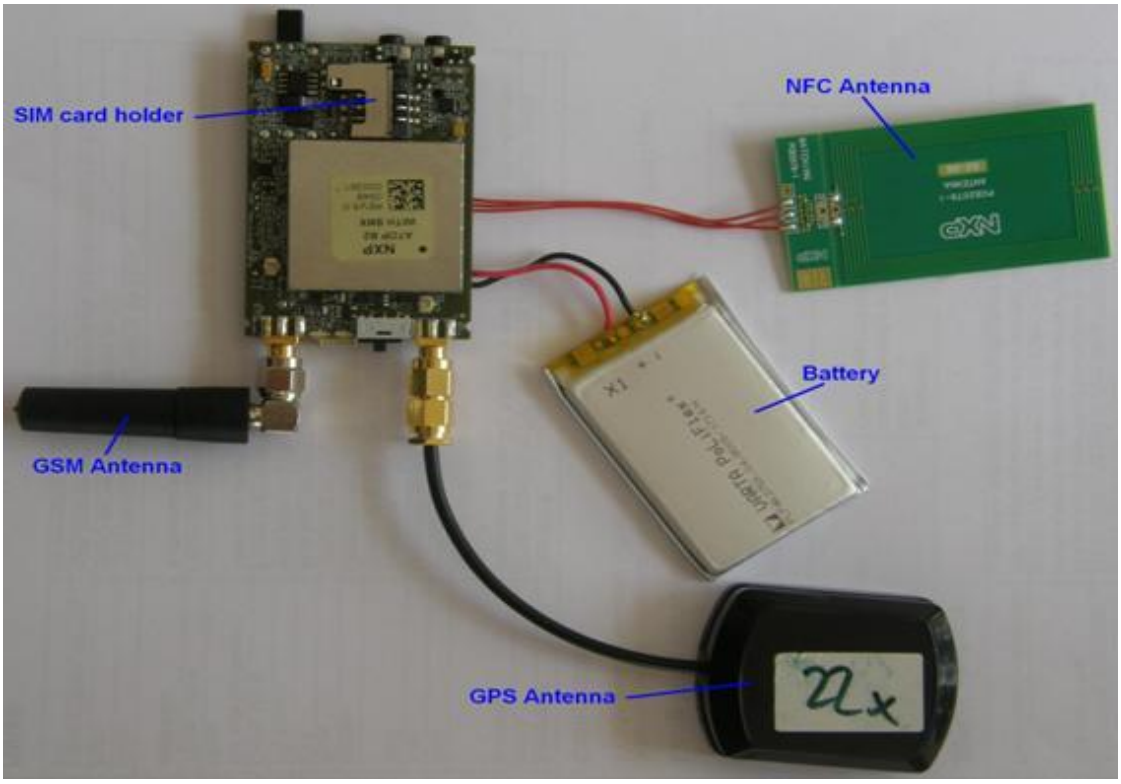
[28] Priscal, Cedric., android-serialport-api. *code.google.com.* [Online] 9 Maart 2010. [Citaat van: 25 augustus 2010.] http://code.google.com/p/android-serialport-api/wiki/Htc.

[29] Android Developers., Android NDK. *developer.android.com.* [Online] Google.com. [Citaat van: 25 august 2010.] http://developer.android.com/sdk/ndk/index.html.

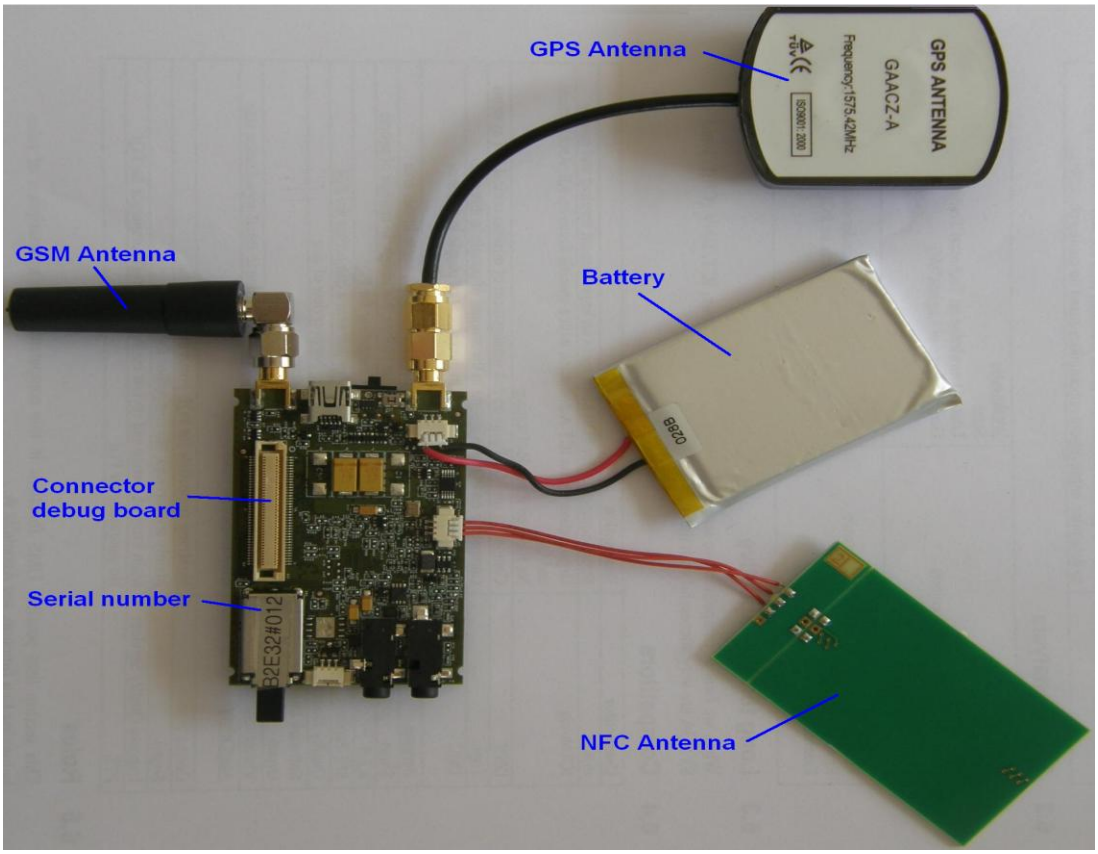
[30] Spits WP1 members., *Overall Architecture.* Rotterdam : sn, 2010.

# Bijlagen

# Bijlage 1. ATOP

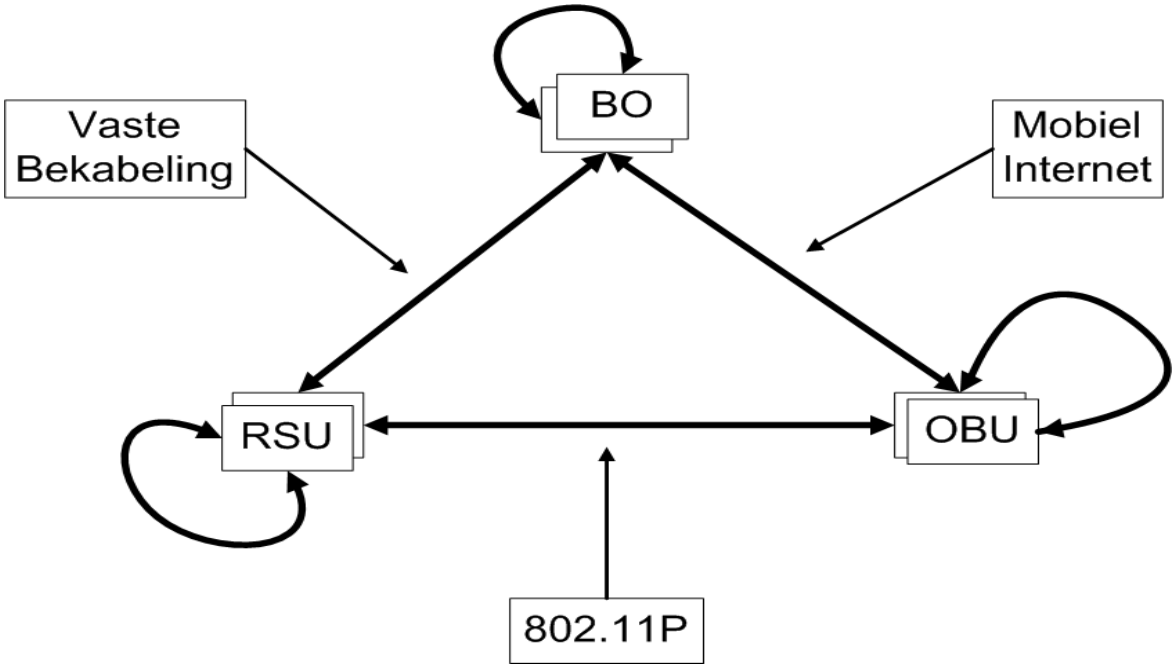



**Figuur 27: ATOP voorkant**



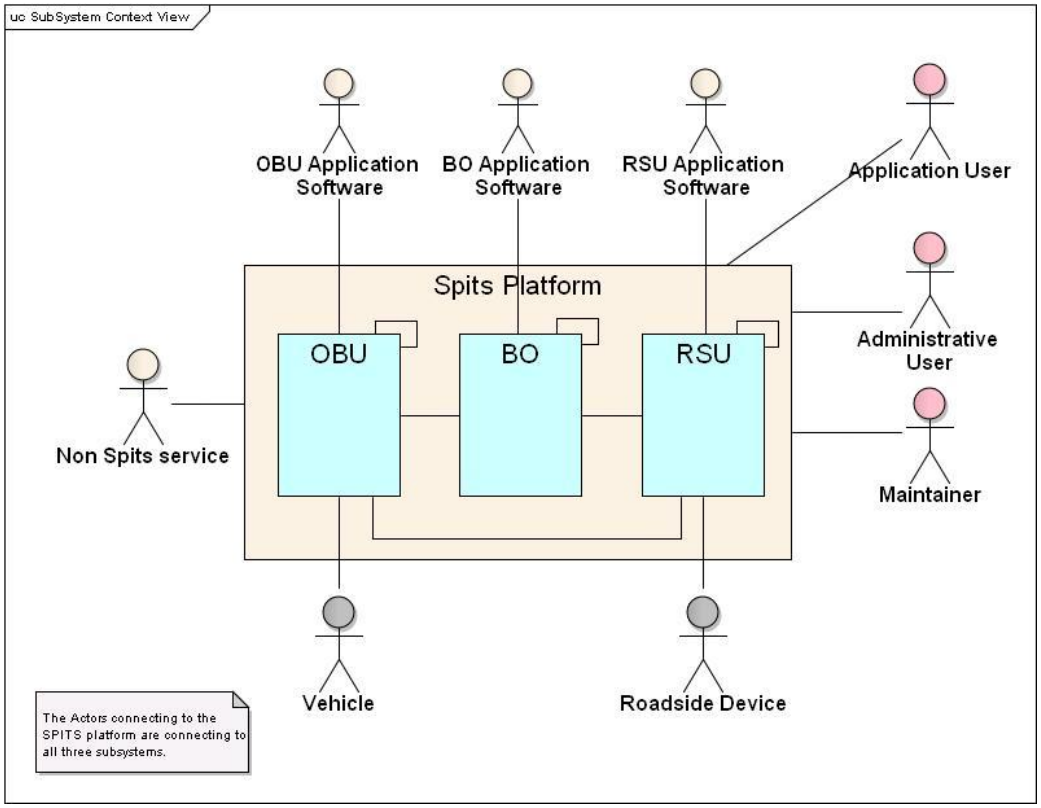

**Figuur 28: ATOP achterkant**

# Bijlage 2. Spits Grote Plaatje



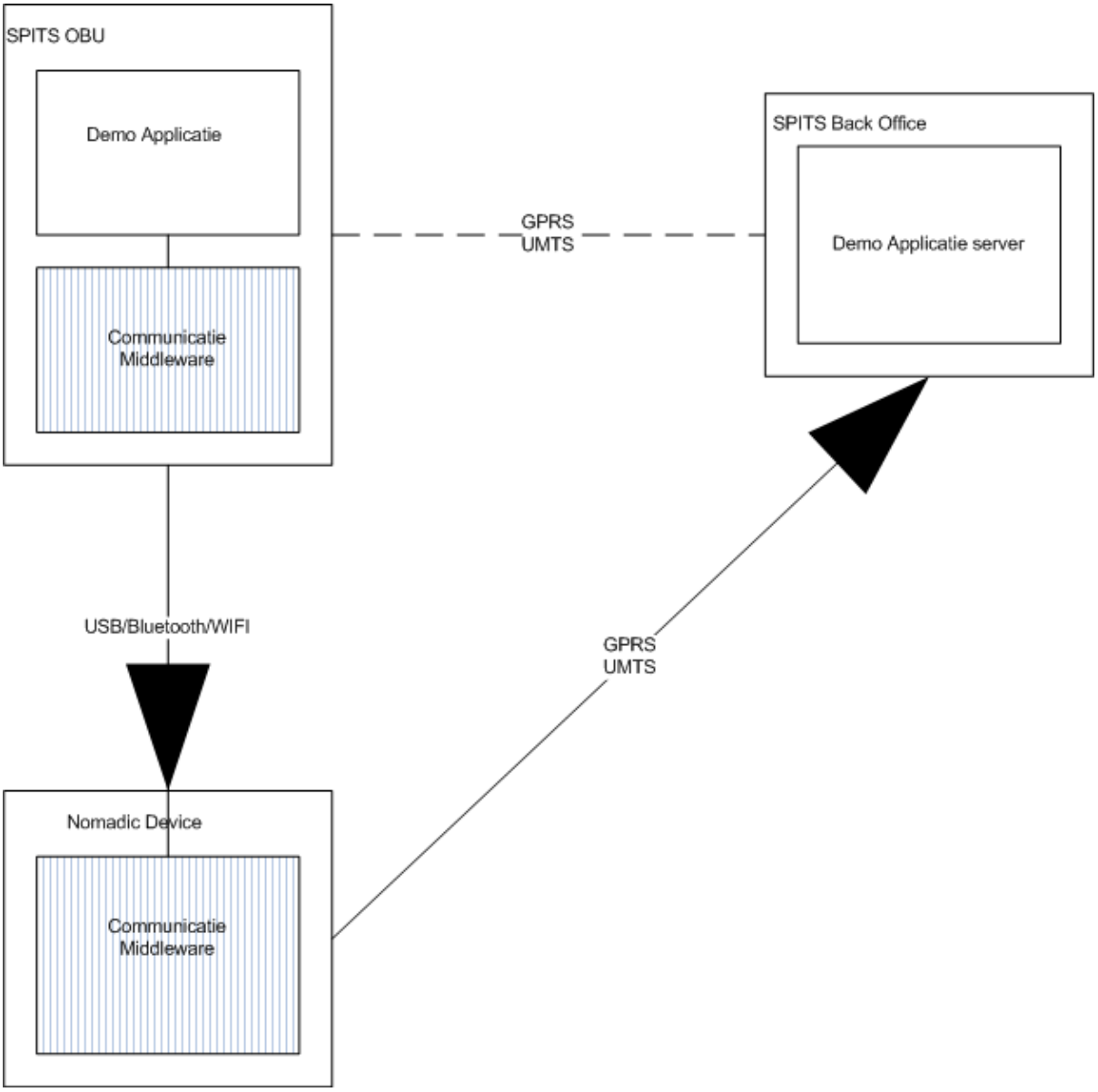

**Figuur 29: Spits Grote Plaatje met protocollen**

# Bijlage 3. Context diagram




**Figuur 30: Context Diagram**

# Bijlage 4. Schets




**Figuur 31: Schets van de oplossing**

# Bijlage 5. Globale Planning (Inceptiefase)

| Week | Inceptiefase | Elaboratiefase | Constructiefase 1 | Constructiefase 2 | Constructiefase 3 |
|---|---|---|---|---|---|
| 1 | • Vision document schrijven | | | | |
| 2 | • Vision document schrijven | | | | |
| 3 | | • CALM documentatie lezen<br>• Uitzoeken werking en protocol USB, BT, WIFI | | | |
| 4 | | • CALM documentatie lezen<br>• Uitzoeken werking en protocol USB, BT, WIFI | | | |
| 5 | | • Uitzoeken werking en protocol USB, BT, WIFI<br>• Eisen opstellen | | | |
| 6 | | • Eisen opstellen<br>• Advies schrijven over gekozen kanaal<br>• Schrijven architectuurrapport | | | |
| 7 | | • Schrijven architectuurrapport | | | |
| 8 | | | • Bouwen Software OBU | | |
| 9 | | | • Bouwen Software OBU | | |
| 10 | | | • Bouwen Software OBU<br>• Testen OBU software<br>• Schrijven ontwerp rapport | | |

| | | | | | |
|---|---|---|---|---|---|
| **11** | | | | • Bouwen software nomadic device | |
| **12** | | | | • Bouwen software nomadic device | |
| **13** | | | | • Bouwen software nomadic device<br>• Testen software OBU en nomadic device<br>• Schrijven ontwerp rapport | |
| **14** | | | | | • Bouwen software alternatief kanaal |
| **15** | | | | | • Bouwen software alternatief kanaal |
| **16** | | | | | • Bouwen software alternatief kanaal<br>• Testen alternatief kanaal<br>• Schrijven ontwerp rapport |
| **17** | Uitloop week | Uitloop week | Uitloop week | Uitloop week | Uitloop week |

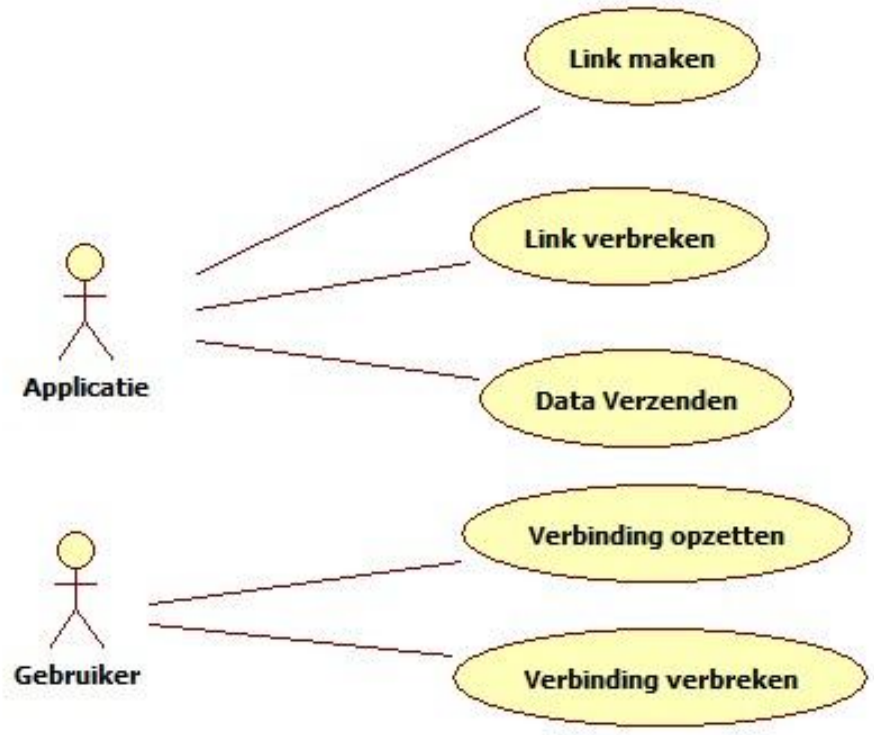
**Tabel 7: Globale Planning (inceptiefase)**

# Bijlage 6. USB Klassenlijst

| Base Class | Descriptor Usage | Description |
|---|---|---|
| 00h | Device | Use class information in the Interface Descriptors |
| 01h | Interface | Audio |
| 02h | Both | Communications and CDC Control |
| 03h | Interface | HID (Human Interface Device) |
| 05h | Interface | Physical |
| 06h | Interface | Image |
| 07h | Interface | Printer |
| 08h | Interface | Mass Storage |
| 09h | Device | Hub |
| 0Ah | Interface | CDC-Data |
| 0Bh | Interface | Smart Card |
| 0Dh | Interface | Content Security |
| 0Eh | Interface | Video |
| 0Fh | Interface | Personal Healthcare |
| DCh | Both | Diagnostic Device |
| E0h | Interface | Wireless Controller |
| EFh | Both | Miscellaneous |
| FEh | Interface | Application Specific |
| FFh | Both | Vendor Specific |

**Tabel 8: USB Klassenlijst**

# Bijlage 7. Use-Case Diagram & Beschrijving




**Figuur 32: Use-case Diagram**

| Naam | **Link maken** |
|---|---|
| **Actor** | **Applicatie** |
| **Pre-Condities** | Nomadic device is aan; Nomadic device is in bereik van de ATOP; De middleware is op de nomadic device aan het draaien. |
| **Post-Condties** | Er is een connectie gemaakt tussen ATOP en nomadic device via USB en tussen nomadic device en BackOffice |
| **Flow of events** | 1. De applicatie op de ATOP wilt data gaan verzenden en meldt dit aan de ATOP.<br>2. De ATOP maakt een connectie met de nomadic device en instrueert de nomadic device om een connectie met de BackOffice te maken.[NO_USB-DEVICE_CONNECTED][BACKOFFICE NOT REACHABLE]<br>3. De applicatie gaat door met use case Data Inpakken |
| **Exepties** | [NO_USB-DEVICE_CONNECTED] Er is geen USB apparaat aan de ATOP gebonden.<br>[BACKOFFICE NOT REACHABLE] de BackOffice is niet bereikbaar. |

| Naam | **Link verbreken** |
|---|---|
| **Actor** | Applicatie |
| **Pre-Condities** | Er is een bestaande connectie tussen nomadic device en ATOP en tussen nomadic device en BackOffice; |
| **Post-Condties** | De link tussen nomadic device en ATOP en tussen nomadic device en BackOffice is verbroken |
| **Flow of events** | 1. De applicatie is klaar met data verzenden en laat dit aan de ATOP weten<br>2. De ATOP stuurt een bericht via de nomadic device naar de BackOffice dat de connectie verbroken moet worden.<br>3. De BackOffice beëindigt de connectie met de nomadic device. |
| **Exepties** | GEEN |

| Naam | **Data verzenden** |
|---|---|
| **Actor** | Applicatie |
| **Pre-Condities** | Er is een bestaande connectie tussen nomadic device en ATOP en tussen nomadic device en BackOffice; De data is ingepakt en is verzendklaar; |
| **Post-Condties** | De data is verzonden. |
| **Flow of events** | 1. De ATOP verzendt de data via de nomadic device naar de BackOffice en meldt de applicatie dat de data verzonden is.[DATA_NOT_ARRIVED]<br>2. De applicatie maakt het volgende pakket klaar om naar de ATOP verzonden te worden om te worden ingepakken. |
| **Exepties** | [DATA_NOT_ARRIVED] De data is verzonden maar is niet aangekomen. De ATOP zal proberen weer een link te leggen met de BackOffice. Zie use case Link maken. |

| Naam | **Verbinding opzetten** |
|---|---|
| **Actor** | Gebruiker |
| **Pre-Condities** | Nomadic device is aan; Nomadic device is in bereik van de ATOP; |
| **Post-Condties** | De nomadic device is klaar om een connectie met de ATOP aan te gaan. |
| **Flow of events** | 1. De gebruiker start de middleware op de nomadic device<br>2. De nomadic device kijkt in de config file naar de instellingen voor de connectie.[NO_CONFIG]<br>3. A) De nomadic device geeft de config scherm weer<br>B) De nomadic device laadt de config file en meldt de ATOP dat ie klaar met opstarten is.<br>4. A) De gebruiker vult de configuratie voor de connectie in<br>5. A) De nomadic device laadt de configuratie in en meldt de ATOP dat ie klaar met het opstartyen is. |
| **Exepties** | [NO_CONFIG] Er is geen config file de nomadic device zal dan de config scherm weergeven en deze opslaan als die ingevuld is. |

| Naam | **Verbinding verbreken** |
|---|---|
| **Actor** | Gebruiker |
| **Pre-Condities** | Nomadic device is aan; Nomadic device is in bereik van de ATOP; Er is een opgezette verbinding. |
| **Post-Condties** | De nomadic device heeft geen verbinding met de ATOP nog de Backoffice. |
| **Flow of events** | 1. De gebruiker geeft aan dat hij de connectie verbinding wilt verbreken.<br>2. De nomadic device verbreekt de verbinding. |
| **Exepties** | GEEN |

# Bijlage 8. Uitleg ATOP (E-mail Han Raaijmakers)

From: Hernandez, Jessie
Sent: woensdag 21 juli 2010 16:04
To: Han Raaijmakers
Subject: programmeren van de USB poort op de atop

Beste Dhr Raaijmakers,

Van Thanh heb ik begrepen dat ik u kan contacten als ik met vragen over de ATOP zit.
Ik ben een student bezig met mijn afstuderen bij Logica in het Spits project.
Ik wil namelijk een Android smartphone aan de ATOP via USB koppelen.
Ik ben nu bezig met programmeren van de ATOP maar ik weet niet hoe ik de USB-poort van de ATOP kan benaderen.

Heeft u misschien ideeën voor mij of library's die ik nodig zou moeten hebben om de ATOP succesvol te programmeren?

Met Vriendelijke Groet,
Jessie Hernandez

---

From: Han Raaijmakers
Sent: woensdag 21 juli 2010 16:24
To: Hernandez, Jessie
Subject: RE: programmeren van de USB poort op de atop

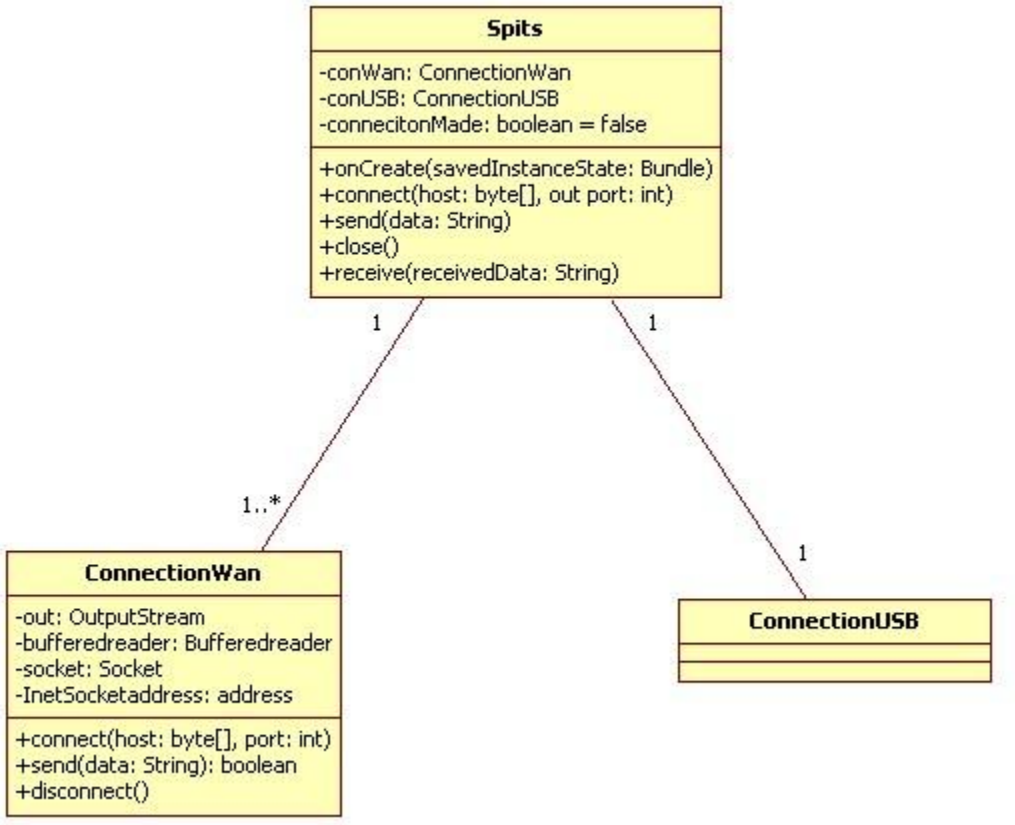
Beste Jessie,

Voor de ATOP is een applicatie beschikbaar (collabnet, Thanh kan je daarbij helpen) waarbij deze zich als USB device gedraagd. Hierbij is de ATOP dan als dual CDC en audio device op USB zichtbaar.
Voor het maken van een connectie is dus op het Andriod platform het volgende nodig:
- USB host
- usb-serial
Via usb serial kun je via terminal programma's (in gewoon leesbaar ASCII) met de ATOP communiceren.
Als dat allemaal goed gaat en je zoekt een verdere uitdaging, kun je usb-audio ook nog toevoegen.

Ik hoop dat dit je vragen beantwoord.

Mvg,
Han Raaijmakers

# Bijlage 9. Klassendiagrammen




**Figuur 33: Klassendiagram nomadic device**

### Spits

Bij initialisatie van de applicatie op het Android platform zal eerst de Spits klasse worden aangeroepen. Deze klasse is een overerving van de klasse Activity. Deze klasse zal eerst een connectie met de ATOP via USB maken. Pas wanneer data binnen komt en klaar is voor het verzenden wordt de connectie met de BackOffice gemaakt.

### ConnectionWan

De ConnectionWan klasse zorgt voor de communicatie tussen nomadic device en BackOffice. Deze klasse heeft vier attributen en drie methodes. Deze worden hieronder uitgelegd.

#### *Attributen*

- Het attribuut out van het type OutputStream wordt gebruikt om de data die vanuit de BackOffice komt te ontvangen.
- bufferedReader van het type BufferedReader is het attribuut die alle data ontvangt en buffert.
- socket van het type Socket wordt gebruikt om de connectie op te zetten
- address van het type InetSocket wordt gebruikt om het IP van de BackOffice op te geven.

***Methodes***

- De methode connect () zorgt voor het initialiseren van de connectie tussen nomadic device en BackOffice. Hiervoor heeft de methode twee parameters nodig. De BackOffice IP als een byte array en de poort waar de connectie gemaakt moet worden.
- send() zorgt ervoor dat de data werkelijk verzonden wordt. Als parameter wordt de data als String verwacht. wanneer de methode is afgelopen wordt er een boolean verstuurd met de waarde true/false. Deze waarde hangt van het feit af of de data intact is aangekomen.
- disconnect() zorgt ervoor dat de connectie met de BackOffice verbroken wordt.

De connectie die opgezet wordt met de BackOffice is een TCP connectie. Dit houdt in dat de transportlaag van het OSI model ervoor zorgt dat de data intact aankomt. Ik heb bewust gekozen dat de BackOffice alsnog een ontvangst bericht terug stuurt naar de nomadic device. Ik heb hiervoor gekozen omdat deze ontvangstbericht door gestuurd moet worden naar de ATOP.

### ConnectionUSB

De conenctionUSB klasse wordt in de constructiefase beschreven. Het is nog niet bekend hoe deze klasse in zijn werking gaat.

# Bijlage 10. HOWTO: maak je een HTC Hero USB host.

## Benodigdheden

- HTC Hero
- Ubuntu (http://www.ubuntu.com/desktop/get-ubuntu/download)
- Android SDK (http://dl.google.com/android/android-sdk_r06-linux_86.tgz)
- HTC hero kernel source code (http://developer.htc.com/)
- Toolchain voor android (http://dl.google.com/android/ndk/android-ndk-r4b-linux-x86.zip Dit is de NDK. De toolchain wordt ook meegeleverd.)
- Kernel patch + config file (http://code.google.com/p/adqmisc/source/browse/#svn/trunk/msm-hsus)
- Flash-rec applicatie (http://zenthought.org/system/files/asset/2/flashrec-1.1.3-20091107-2.apk)
- Recovery rom (http://rapidshare.com/files/360227978/recovery-RA-hero-v1.6.2.img)
- HTC android Kitchen (http://forum.xda-developers.com/showthread.php?t=633246)
- JAVA 1.6
- HTC hero default ROM (ftp://xda:xda@ftp.xda-developers.com/Hero/Official-ROMs/RUU_Hero_HTC_WWE_2.73.405.66_WWE_test_signed_NoDriver.exe)*
- Connectbot (verkrijgbaar in de Andriod store)
- Veel geduld

Dit document beschrijft hoe je de kernel van een HTC Hero als USB host kan laten optreden.

**Tijd: 60-90 minuten**
**Moeilijkheidsgraad: Hoog**
**Benodigde kennis/ervaring: ervaring met het Linux besturingsysteem en kennis van Terminal functies binnen Linux**

**Stap 1:**
Download alle benodigdheden.

**Stap 2:**
Pak de Android SDK en NDK uit. (je krijgt een folder met de naam android-sdk-linux_86)

**Stap 3:**
zet de Android SDK tools folder in het systeem variabele PATH *
`Export PATH = $PATH:<pad naar de Android SDK>/tools`

**Stap 4:**
Zet de Android Toolchain in het systeem variabele PATH *
`Export PATH = $PATH:<pad naar de android ndk>/build/prebuilt/linux-x86/arm-eabi-4.2.1/bin`

**Stap 5:**
Maak een folder in je home directory waar je alle tools en source code in kwijt kan
`mkdir ~/myandroid.`

**Stap 6:**
Pak de kernel in de myandroid folder uit. Je hoort dan een folder in de myandroid folder te krijgen met de source code.

**Stap 7:**
Kopieer de patch file in de root van de kernel.
Ga daarna naar de kernel root
`Cd ~/myandroid/<kernel folder>`

**stap 8:**
patch the kernel
`patch -p1 <  android-msm-2.6.27-hero-v23.patch`

**stap 9:**
Kopieer de config file in de root van de kernel folder.
Verander de naam van de file in .config.
`cp android-msm-2.6.27-hero-config .config`

**Stap 10:**
compileer de kernel
`make ARCH=arm CROSS_COMPILE=arm-eabi-`

Tijd voor een kopje koffie dit duurt zeker een half uur voordat het klaar is.

**stap 11:**
Als alles goed is gegaan heb je nu geen foutmeldingen en is de kernel gecompileerd.

**Stap 12:**
maak de telefoon klaar voor eigen ROM.

Zet de recovery rom & de apk op je sd kaart. Sluit je toestel aan op de PC en kopieer de bestanden naar je SD kaart.

Zorg nu dat je telefoon zo ingesteld staan dat je van onbekende bronnen software mag installeren. (Instellingen -> Applicaties -> onbekende bronnen aanvinken)

Installeer de apk door een filebrowser uit de Android market te download (bijvoorbeeld Linda Filemanager). Navigeer vervolgens naar je SD kaart en klink op flashrec-1.1.3-20091107-2.apk

Open de flash-rec applicatie die je zojuist geïnstalleerd hebt.

Klik nu op de knop “back-up recovery image” dit maakt een backup van de huidige HTC bootloader.

Zodra dit gedaan is, kan je de bootloader vervangen. Zorg dat de telefoon niet meer verbonden is met je PC en tik het volgende in het witte tekst vakje: /sdcard/recovery-RA-hero-v1.6.2.img (LET OP! Hoofdlettergevoelig, maak geen tik fouten, dit kan je bootloader beschadigen)

Nu is de bootloader vervangen, en is je telefoon geroot!

**Stap 13:**
Pak de kitchen uit in de myandroid folder.

**Stap 14:**
Zet de ROM file (een zip bestand) in de original_update folder van de kitchen.**

**stap 15:**
Maak je working folder aan.
Ga naar de kitchen folder en tik `./menu`

```
cd ~/myandroid/<kitchen folder>  [enter]
./menu
```

Kies menu 1

en dan kies menu 2 om root rechten aan de kernel te voegen.

Bij het compileren zijn er een aantal modules gemaakt. Deze moeten ook meegestuurd worden samen bij het klaar maken van de kernel anders herkent de hero deze niet en zal dit niet werken.

**Stap 16:**
in de root van de kernel folder is er een bestand `modules.order`. Maak deze open en dan zie je alle modules die gemaakt zijn. Je hoeft niet alle modules mee te nemen in de ROM. Alleen de modules die je wilt gaan gebruiken. Kopieer deze in de `system/lib/modules` folder in de working folder van je kitchen.

De modules die nodig zijn voor USB host en USB slave zijn

- ehci-hcd.ko → USB Host
- f_msm_hsusb.ko → USB Slave

**Stap 17**
in de kitchen:
Ga naar advance options (menu 10)
Kies optie 15. extract kernel + ramdisk from working folder

**Stap 18:**
Kopieer de zImage naar de BOOT-EXTRACTED folder. (te vinden in `kernel folder/arch/arm/boot)`

Delete de boot.img-kernel file

Geef de zImage de naam boot.img-kernel

**Stap 19:**
Pak de nieuwe ROM in
Kies optie 16. Build new boot.img from kernel + ramdisk (for previous option)

**Stap 20:**
Maak de ROM.
Ga terug naar het hoofdmenu. Tik 0 in
kies optie 99. Build ROM from working folder.

**Stap 21:**
Kopieer het zip bestand uit OUTPUT_ZIP en plaats deze op de sdcard van de telefoon.

**Stap 22:**
Maak de telefoon uit.
Houd de home knop in terwijl je de aan knop drukt.

**Stap 23:**
Kies flash zip from sdcard.
En kies het zip bestand
Bevestig je keuze door op de home knop te drukken.

**Stap 24:**
vanaf een terminal (connectbot te vinden in de android store gratis) moet je de module nu inladen.

Open connectbot en wordt root `su`

**stap 25:**
ga naar de modules folder
`cd /system/lib/modules`

**stap 26:**
laad de module.
`insmod ehci-hcd.ko`

**stap 27:**
steek een toetsenboard met een Y-kabel en dan moet het werken. De Y-kabel is nodig om de USB apparaat van stroom te voorzien. Een USB Y-kabel kunt vanaf het internet kopen.

## Voor meer informatie kun je de volgende website raadplegen

- http://developer.android.com/sdk/index.html- SDK problemen
- http://developer.android.com/sdk/ndk/index.html - NDK problemen (Toolchain)
- http://honeypod.blogspot.com/2007/12/compile-android-kernel-from-source.html- Compileer problemen
- http://www.androidworld.nl/20048/howto-htc-hero-rooten-en-android-2-1-installeren/ - Root problemen
- http://theunlockr.com/2010/03/26/how-to-create-your-own-custom-rom-for-android-part-1-setting-up-the-kitchen/%20-%20ROM%20problemen- ROM problemen
- http://theunlockr.com/2010/04/15/how-to-create-your-own-custom-rom-for-android-part-2-creating-your-first-rom/ - ROM problemen
- http://www.google.com/- Algemene problemen

## Excepties

* Deze variabele zal worden gereset na het herstarten van het systeem.

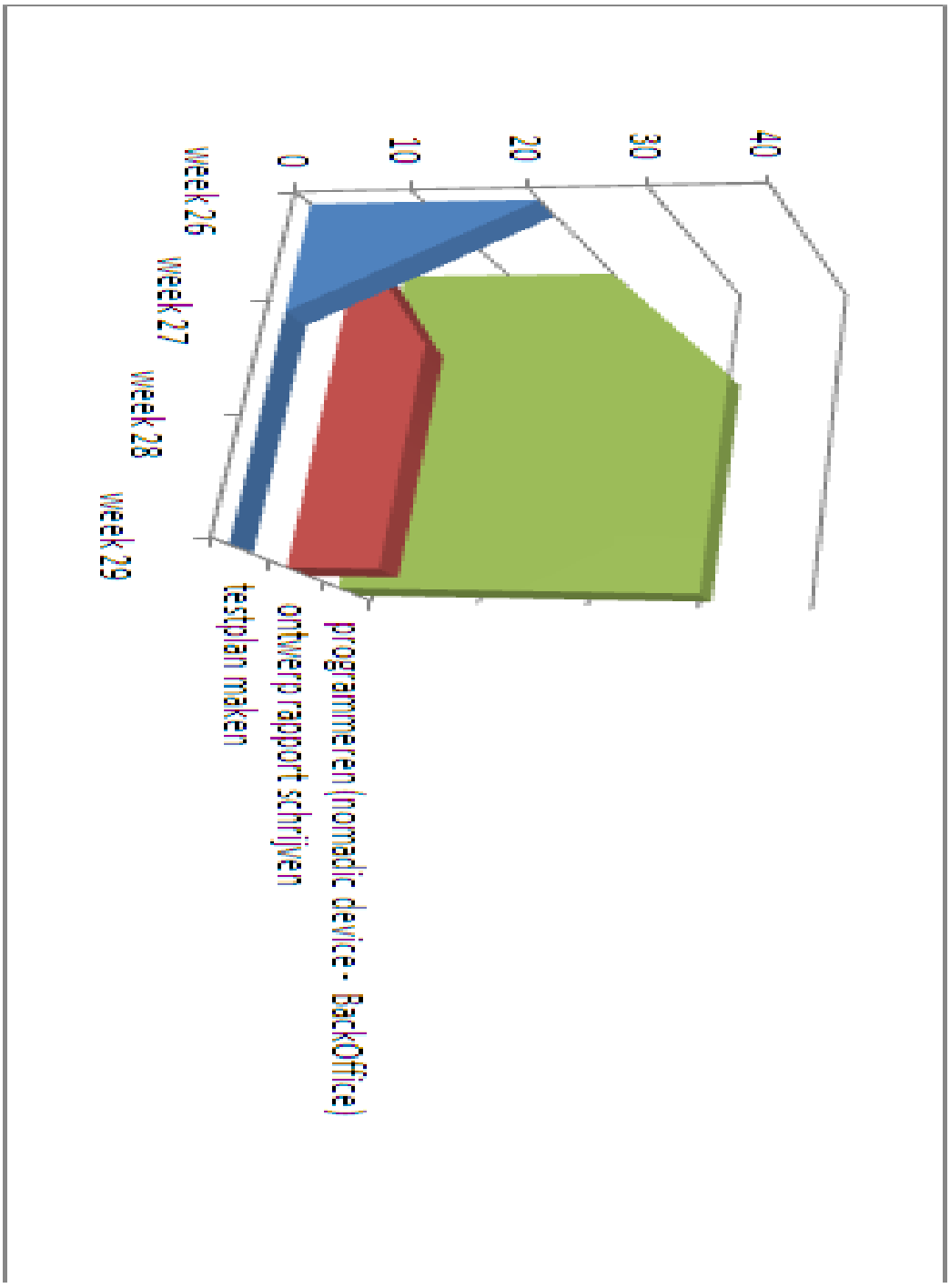
**Als de ROM een .EXE als extensie heeft voer het volgende eerst uit.

A. In Windows. Voer de .EXE uit tot wanneer je de eerste vraag krijgt. Stop met de installatie maar sluit de installatie nog niet af.
B. Ga naar Start Run en tik %TEMP%
C. Wanneer de folder open is zoek naar Rom.zip
D. Kopieer deze op een USB-stick en ga verder met stap 19

# Bijlage 11. Globale Planning (elaboratiefase)

| Week | Inceptiefase | Elaboratiefase | Constructiefase 1 | Constructiefase 2 |
|---|---|---|---|---|
| 3 | | CALM documentatie lezen<br>Uitzoeken werking en protocol USB, BT, WIFI | | |
| 4 | | CALM documentatie lezen<br>Uitzoeken werking en protocol USB | | |
| 5 | | Uitzoeken werking en protocol USB, Eisen opstellen | | |
| 6 | | Nomadic device gereed maken voor programmeren | | |
| 7 | | Nomadic device gereed maken voor programmeren | | |
| 8 | | Nomadic device gereed maken voor programmeren | Bouwen Software Nomadic device - BackOffice | |
| 9 | | Nomadic device gereed maken voor programmeren | Bouwen Software Nomadic device - BackOffice | |
| 10 | | Nomadic device gereed maken voor programmeren | Testen OBU software<br>Schrijven ontwerp rapport | |
| 11 | | Nomadic device gereed maken voor programmeren | | |
| 12 | | Nomadic device gereed maken voor programmeren | | |
| 13 | | Nomadic device gereed maken voor programmeren | | |
| 14 | | Nomadic device gereed maken voor programmeren | | Bouwen software nomadic device |
| 15 | | Schrijven van architectuur document | | Bouwen software nomadic device |
| 16 | | Schrijven van architectuur document | | Bouwen software nomadic device<br>Testen software OBU en nomadic device<br>Schrijven ontwerp rapport |
| 17 | Uitloop week | Uitloop week | Uitloop week | Uitloop week |

# Bijlage 12. Iteratieplan Constructiefase 1




Figuur 34: iteratieplan constructiefase 1

# Bijlage 13. testgevallen

## Testgevallen Nomadic device – BackOffice (iteratie 1)

### Error Guessing

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan. |
| Taken: | 1. Start de applicatie op de mobiele apparaat.<br>2. Druk op de knop verzenden. |
| Verwachtte uitvoer: | De tekst “Welcome to the spits open platform” verschijnt op het scherm bij de BackOffice er verschijnt een label onder de button met de juist verstuurde tekst als hoofdletters. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

### Real-life test

#### Profielschets

Om de real-life test zo realistisch mogelijk te maken is er profielschets vereist. De profielschets zal de situatie beschrijven waarin getest zal worden. In de volgende paragraaf vind u de situatie waarin de real-life test zal worden uitgevoerd.

De test zal mobiel moeten plaatsvinden doordat de OBU in een auto zal worden geplaatst. Er wordt echter gekozen om de test toch niet in een auto uit te voeren omdat dit geen toegevoegde waarde heeft op de test. De test zal representatief zijn omdat de data vanaf de OBU tot het mobiele apparaat via USB verloopt en dus geen interferentie ondervindt.

Bij de connectie tussen het mobiele apparaat en de BackOffice zal de sessie niet onderbroken worden bij het veranderen van zendmast. De IP-adressen zijn namelijk statische adressen. De implementatie van het mobiele internet is echter per provider verschillend en is dus provider afhankelijk of het IP dynamisch of statisch is.

De tests worden uitgevoerd bij T-Mobile als provider. Na contact op te nemen met de technische dienst van T-Mobile is gebleken dat T-Mobile gebruik maakt van statische IP-adressen. Dit betekent dat sessies gewoon open blijven.

Als omgeving wordt daarom gekozen om de tests in de ontwikkel omgeving te doen. Een andere reden voor deze keuze is de mogelijkheid om de debug board van de OBU te koppelen om zo ook de systeem berichten van de OBU te kunnen lezen.

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan. |
| Taken: | 1. Start de applicatie op de mobiele apparaat.<br>2. Druk 5x op de knop verzenden. |
| Verwachtte uitvoer: | De tekst “Welcome to the spits open platform” verschijnt5 maal op het scherm bij de BackOffice er verschijnt een label onder de button met de juist verstuurde tekst als hoofdletters. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

# Testgevallen communicatielink (Iteratie 2)

## Error Guessing

Als mogelijke zwakke plekken in de communicatielink heb ik bedacht dat het verzenden en ontvangen van data via het USB kanaal problemen kan opleveren. Een tweede zwakke plek is de foutafhandeling wanneer er wat mis gaat in de applicatie.

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 1. Start de applicatie op de mobiele apparaat.<br>2. Sluit de stroom stekker aan.<br>3. Als er gevraagd wordt om root rechten kies “allow”* |
| Verwachtte uitvoer: | De data op de ATOP binnengekomen met als verandering dat de gestuurde data als hoofdletters terug is gestuurd. |
| Werkelijke uitvoer | |
| Opmerkingen: | |
| Naam: | Connectie verbreken |
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 1. Start de applicatie op de mobiele apparaat. |

| | |
|---|---|
| | 2. Sluit de stroom stekker aan.<br>3. Als er gevraagd wordt om root rechten kies “allow”*<br>4. Na 5 secondes trek de stekker uit het mobiele apparaat. |
| Verwachtte uitvoer: | Er verschijnt een melding op de mobiele apparaat met de boodschap dat de data niet kon worden verzonden. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

### Testgevallen real-life test

| Naam: | De applicatie stuurt de data synchroon, er moet altijd een response vanuit de ATOP naar de applicatie. |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 1. Start de applicatie op de mobiele apparaat.<br>2. Sluit de stroom stekker aan.<br>3. Als er gevraagd wordt om root rechten kies “allow”*<br>4. Kijk op de pc in het programma spyTracer naar het outputscherm. |
| Verwachtte uitvoer: | De applicatie op de OBU blijft ‘hangen’ tot wanneer er data terug gestuurd is vanaf de BackOffice/mobiele apparaat. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

| Naam: | De link moet automatisch opgezet worden zonder input van de gebruiker. |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 1. Start de applicatie op de mobiele apparaat.<br>2. Sluit de stroom stekker aan.<br>3. Als er gevraagd wordt om root rechten kies “allow”* |
| Verwachtte uitvoer: | De connectie wordt opgezet en de data wordt verstuurd van OBU naar mobiele apparaat naar BackOffice en terug. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

*er kan ook gekozen worden voor “always allow”.

# Bijlage 14. Testgevallen & resultaat

## Testgevallen Nomadic device – BackOffice (iteratie 1)

### Error Guessing

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan. |
| Taken: | 3. Start de applicatie op de mobiele apparaat.<br>4. Druk op de knop verzenden. |
| Verwachtte uitvoer: | De tekst “Welcome to the spits open platform” verschijnt op het scherm bij de BackOffice er verschijnt een label onder de button met de juist verstuurde tekst als hoofdletters. |
| Werkelijke uitvoer | De label presenteert de tekst zoals het hoort en de BackOffice ontvangt de data ook intact. |
| Opmerkingen:<br><br>geen | |

### Real-life test

#### Profielschets

Om de real-life test zo realistisch mogelijk te maken is er profielschets vereist. De profielschets zal de situatie beschrijven waarin getest zal worden. In de volgende paragraaf vindt u de situatie waarin de real-life test zal worden uitgevoerd.

De test zal mobiel moeten plaatsvinden doordat de OBU in een auto zal worden geplaatst. Er wordt echter gekozen om de test toch niet in een auto uit te voeren omdat dit geen toegevoegde waarde heeft op de test. De test zal representatief zijn omdat de data vanaf de OBU tot het mobiele apparaat via USB verloopt en dus geen interferentie ondervindt.

Bij de connectie tussen het mobiele apparaat en de BackOffice zal de sessie niet onderbroken worden bij het veranderen van zendmast. De IP-adressen zijn namelijk statische adressen. De implementatie van het mobiele internet is echter per provider verschillend en is dus provider afhankelijk of het IP dynamisch of statisch is.

De tests worden uitgevoerd bij T-Mobile als provider. Na contact op te nemen met de technische dienst van T-Mobile is gebleken dat T-Mobile gebruik maakt van statische IP-adressen. Dit betekent dat sessies gewoon open blijven.

Als omgeving wordt daarom gekozen om de tests in de ontwikkelomgeving te doen. Een andere reden voor deze keuze is de mogelijkheid om de debug board van de OBU te koppelen om zo ook de systeem berichten van de OBU te kunnen lezen.

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan. |
| Taken: | 1. Start de applicatie op de mobiele apparaat.<br>2. Druk 5x op de knop verzenden. |
| Verwachtte uitvoer: | De tekst “Welcome to the spits open platform” verschijnt5 maal op het scherm bij de BackOffice er verschijnt een label onder de button met de juist verstuurde tekst als hoofdletters. |
| Werkelijke uitvoer | De label presenteert de data zoals het hoort en de BackOffice heeft de data ook 5 maal ontvangen. |
| Opmerkingen:<br><br>geen | |

# Testgevallen communicatielink (Iteratie 2)

## Error Guessing

Als mogelijke zwakke plekken in de communicatielink heb ik bedacht dat het verzenden en ontvangen van data via het USB kanaal problemen kan opleveren. Een tweede zwakke plek is de foutafhandeling wanneer er wat mis gaat in de applicatie.

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 1. Start de applicatie op de mobiele apparaat.<br>2. Sluit de stroom stekker aan.<br>3. Als er gevraagd wordt om root rechten kies “allow”* |
| Verwachtte uitvoer: | De data op de ATOP binnengekomen met als verandering dat de gestuurde data als hoofdletters terug is gestuurd. |
| Werkelijke uitvoer | StreamCorruptedException/ InvalidClassException |
| Opmerkingen:<br><br>Dit komt door het deserialiseren van het object. Doordat de data niet goed aankomt door nog onbekende redenen. | |

| Naam: | Connectie verbreken |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 5. Start de applicatie op de mobiele apparaat.<br>6. Sluit de stroom stekker aan.<br>7. Als er gevraagd wordt om root rechten kies “allow”*<br>8. Na 5 seconden trek de stekker uit het mobiele apparaat. |
| Verwachtte uitvoer: | Er verschijnt een melding op de mobiele apparaat met de boodschap dat de data niet kon worden verzonden. |
| Werkelijke uitvoer | Klopt. Er verschijnt een boodschap dat de connectie niet gelukt is. |
| Opmerkingen:<br><br>De boodschap verschijnt ook als de connectie wel gelukt is maar de data niet verzonden is | |

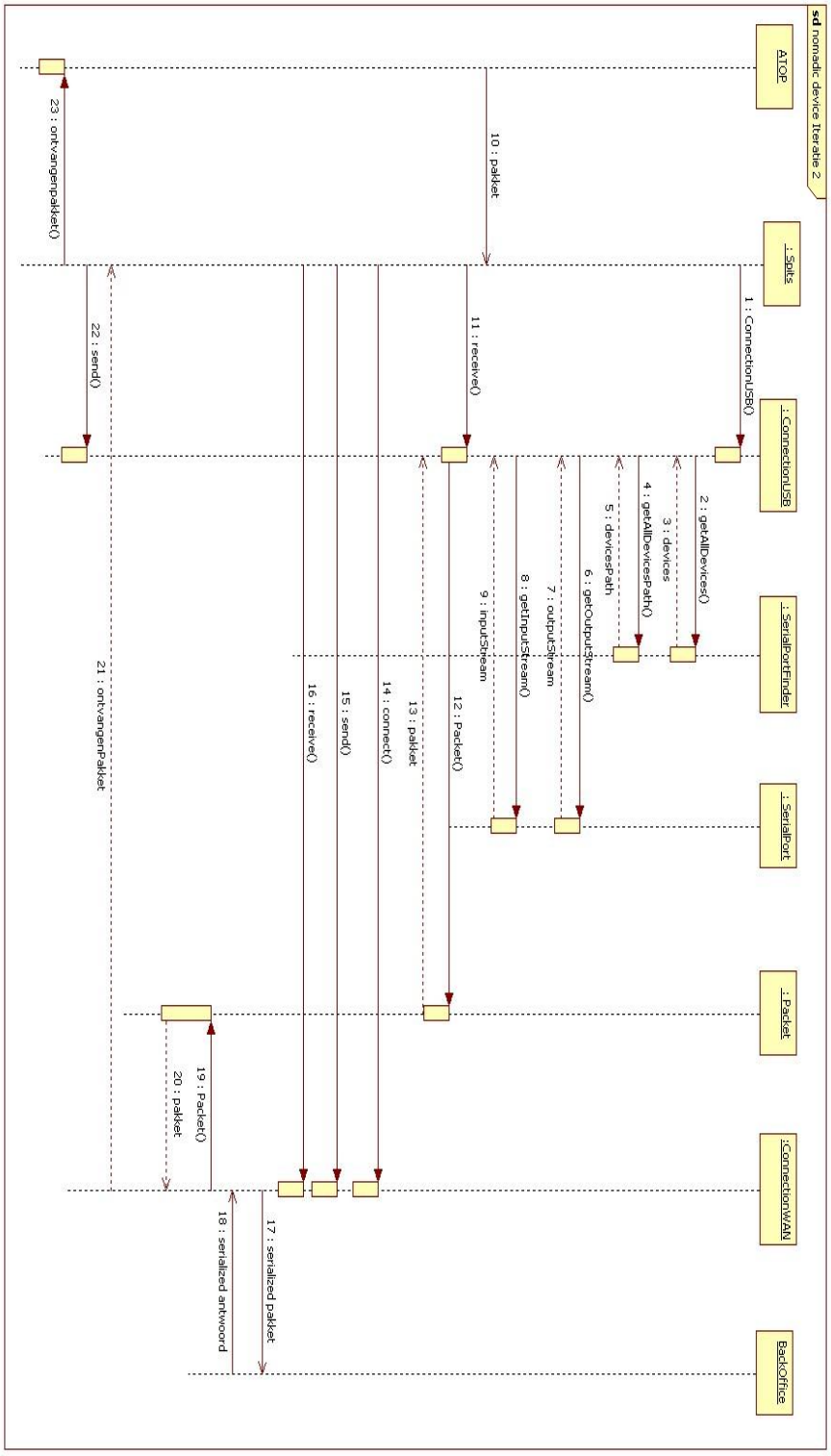
## Testgevallen real-life test

| Naam: | De applicatie stuurt de data synchroon, er moet altijd een response vanuit de ATOP naar de applicatie. |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 5. Start de applicatie op de mobiele apparaat.<br>6. Sluit de stroom stekker aan.<br>7. Als er gevraagd wordt om root rechten kies “allow”*<br>8. Kijk op de pc in het programma spyTracer naar het outputscherm. |
| Verwachtte uitvoer: | De applicatie op de OBU blijft wachten tot wanneer er data terug gestuurd is vanaf de BackOffice/mobiele apparaat. |
| Werkelijke uitvoer | De applicatie blijft wachten maar er komt geen data terug |
| Opmerkingen:<br><br>de data komt niet goed aan op het Android toestel. Dit komt nog door onbekende redenen. | |

| Naam: | De link moet automatisch opgezet worden zonder input van de gebruiker. |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 4. Start de applicatie op de mobiele apparaat.<br>5. Sluit de stroom stekker aan.<br>6. Als er gevraagd wordt om root rechten kies “allow”* |
| Verwachtte uitvoer: | De connectie wordt opgezet en de data wordt verstuurd van OBU naar mobiele apparaat naar BackOffice en terug. |
| Werkelijke uitvoer | De connectie wordt automatisch opgezet. Maar de data blijft bij de OBU hangen. |
| Opmerkingen:<br><br>De data komt niet goed aan op het Android toestel door nog onbekende redenen. | |

*er kan ook gekozen worden voor “always allow”.

# Bijlage 15. Sequentiediagram Android toestel Iteratie 2




**Figuur 35: Sequentiediagram android toestel Iteratie 2**

# Opgeleverde Tussentijds Documenten

# Afstudeerplan

## Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

*Afstudeerblok*: 2010-1.2
*Startdatum*: 10 mei 2010
*Einddatum:*24 september 2010
*Inleverdatum:* 24 september 2010

*Studentnummer*: 20050537
*Achternaam*: dhr Hernandez (*) weghalen niet van toepassing
*Voorletters*: J J J
*Roepnaam*: Jessie
*Adres*: Stamkartplein 278
*Postcode*:2521ES
*Woonplaats*: Den Haag
*Telefoon*: - *Mobiel*: 0620280558

*Opleiding*: Technische Informatica
*Locatie*: Den Haag (*) weghalen niet van toepassing
*Variant*: voltijd

*Naam studieloopbaanbegeleider*: Ron van Neijhof
*Naam begeleider/examinator*: Hans Witkamp
*Naam expert/examinator*: Cobie van der Hoek

*Naam bedrijf*: Logica
*Afdeling bedrijf*: Working Tomorrow Rotterdam
*Bezoekadres bedrijf*: George Hintzenweg 89
*Postcode bezoekadres*: 3068 AX
*Postbusnummer*: 8566
*Postcode postbusnummer*: 3009 AN
*Plaats*: Rotterdam
*Telefoon bedrijf*: 010-2537000
*Telefax bedrijf*: 010-2537001
*Internetsite bedrijf*: www.logica.com

*Achternaam opdrachtgever*: dhr de Neef (*) weghalen niet van toepassing
*Voorletters opdrachtgever: A.J.*.
*Titulatuur opdrachtgever*: mr
*Functie opdrachtgever*: Interim Manager
*Doorkiesnummer opdrachtgever*: 030-6027492/0622417451
*Email opdrachtgever*: bert.de.neef@logica.com

*Achternaam bedrijfsmentor*: dhr. De Jong (*) weghalen niet van toepassing
*Voorletters bedrijfsmentor*: Ferry
*Titulatuur bedrijfsmentor*: -
*Functie bedrijfsmentor*: Technical Consultant
*Doorkiesnummer bedrijfsmentor*: +31 (0) 88 56 47584
*Email bedrijfsmentor*: ferry.de.jong@logica.com

*Doorkiesnummer afstudeerder:*
*Bedrijfsemail afstudeerder*:
*Functie afstudeerder (deeltijd/duaal)*:

## Opdrachtomschrijving

Titel afstudeeropdracht: SPITS in-vehicle koppelingsprotocol voor een mobiel apparaat.

1. Bedrijf
De student zal zijn afstudeeropdracht bij Logica Nederland uitvoeren. Binnen dit bedrijf is er een project met de naam SPITS. Dit project ontwerpt mogelijke prototypes die ingezet kunnen worden om het autorijden veiliger te maken.

2. Aanleiding
Aanleiding voor deze opdracht is de ontwikkeling binnen het SPITS project van concept producten voor 3$^{de}$ partijen die diensten leveren voor autorijders. Voorbeelden hiervan zijn verzekeraars, leasemaatschappijen, autoverhuurbedrijven en andere mogelijke wagenparkbeheerders. SPITS wil oplossingen wegzetten waarmee dergelijke partijen diensten kunnen worden aangeboden zoals een remote-diagnostics functionaliteit (het op afstand monitoren van de toestand van een voertuig), een emergency-call functionaliteit (het automatisch inschakelen van hulpdiensten in geval van een ongeval), een stolen-vehicle functionaliteit (het tracken en tracen van gestolen auto's), of een pay-as-you-drive functionaliteit (het gebruik maken van het rijgedrag voor het bepalen van kosten zoals huurprijs, leaseprijs, of een verzekeringspremie). Hierbij willen we gebruik maken van een vast kastje in de auto: de OBU (On Board Unit). De OBU is een in-vehicle device (of mogelijk combinatie van meerdere apparaten), welke deel uitmaakt van het SPITS platform en waarop SPITS applicaties kunnen draaien. Binnen SPITS wordt momenteel gewerkt aan de ontwikkeling van dergelijke in-vehicle systemen, welke beschikken over standaard, vaste, communicatiekanalen, met de daarbij behorende communicatiekosten. De On Board Unit beschikt over mogelijkheden om te communiceren met de SPITS back-office, maar het is nog onduidelijk wat de kosten hiervan zullen zijn omdat de OBU nog een experimenteel product is.

3. Probleemstelling
Om onafhankelijk te kunnen zijn van een specifieke Telecom leverancier, willen we de mogelijkheid hebben om ook via een ander kanaal datacommunicatie te realiseren. Zonder gebruik te maken van de communicatie mogelijkheden van de OBU.

4. Doelstelling van de afstudeeropdracht.
Het realiseren van een proof of concept van een middlewarelaag om datacommunicatie van de OBU met de back-office tot stand te brengen via een nomadic device

5. Resultaat.
Het resultaat van de opdracht moet de volgende punten bevatten:
- Uitgewerkt software-protocol waarmee mobiele apparaten eenvoudig en dynamisch kunnen koppelen met het OBU. Hierbij moet rekening gehouden worden met het feit dat er meerdere nomadice devices of OBU's betrokken kunnen zijn bij de koppeling.
- Middleware voor het beschikbaar maken van een communicatiekanaal op een nomadic device aan een SPITS OBU, bestaande uit software op de OBU en software op de nomadic device.
- Werkend prototype waarbij de OBU communiceert met de SPITS back-office, via een nomadic device

6. Uitgangssituatie aan de hand van:
   a. Beschikbare noodzakelijke software (*indien van toepassing*)

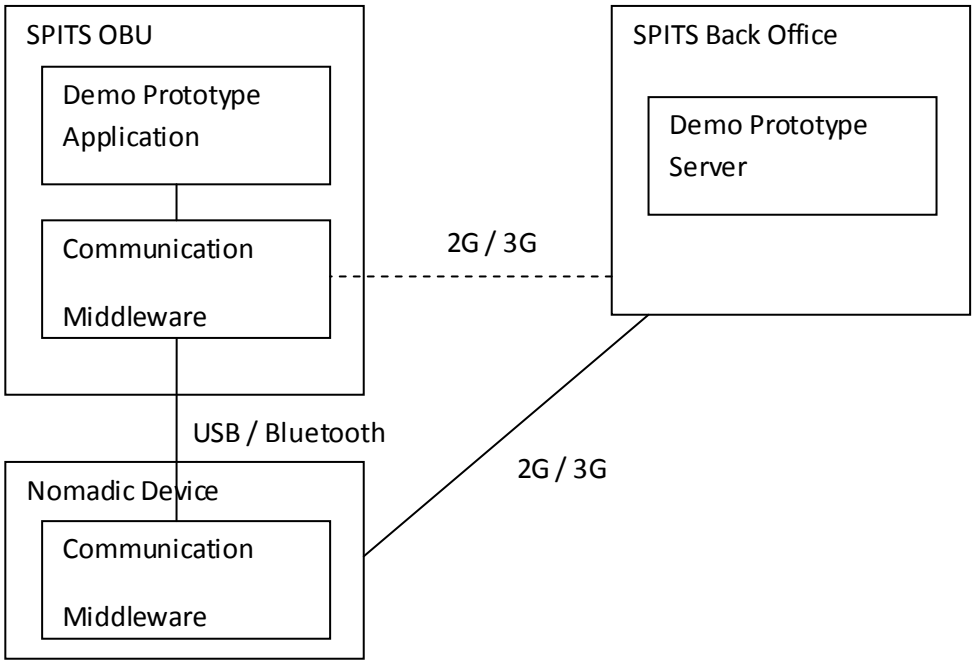
      De software development kit van het Android besturing systeem

   b. Beschikbare noodzakelijke hardware (*indien van toepassing*)
      Mobiele telefoon Android Developers Phone.
      OBU met standaard communicatie kanalen.
      SPITS Back-office

   c. Reeds opgeleverde relevante documenten
      CALM documentatie

   d. Aanwezige ideeën
      Een vergelijkbaar stuk middleware is reeds eerder ontwikkeld in het CVIS project (Cooperative Vehicle Infrastucture Systems): het CALM concept (Communications Access for Land Mobiles). Echter, het gebruik van nomadic devices is hierin niet uitgewerkt en de ontwikkelde software draait op CVIS pc's en niet op de SPITS OBU's.

      Het huidige idee van de opzet van de software en netwerkcommunicatie voor deze opdracht is in onderstaande figuur weergegeven.




Er kan gebruik gemaakt worden van standaard kanalen zoals het datanetwerk van een Telecom leverancier, en voor communicatie met de OBU bijv USB, WIFI en Bluetooth. Deze laag zal deels op de nomadic device en deels op de OBU gemaakt worden.

7. Werkzaamheden aan de hand van:
   a. Te hanteren methodieken
      De uitwerking van de software protocol, het ontwerpen van de middleware en een werkend prototype die het communicatiekanaal aantoont worden allemaal met de software ontwikkelmethode RUP ontworpen

   b. Uit te voeren activiteiten
      Inceptiefase
      - schrijven van vision document

      Elaboratiefase
      - CALM documentatie lezen
      - uitzoeken werking en protocol USB, Bluetooth, WIFI.
      - eisen opstellen
      - adviseren over de te gebruiken communicatie kanaal.
      - Schrijven van architectuur rapport

      Constructiefase 1
      - bouwen van de software voor de OBU (USB of Bluetooth)
      - Testen van de software voor de OBU
      - Schrijven van het ontwerp rapport

      Constructiefase 2
      - bouwen van de software voor de nomadic device
      - Testen van de software voor de nomadic device en OBU samen
      - Schrijven van het ontwerp rapport

      Constructiefase 3
      - bouwen van de software voor alternatief communicatie kanaal (als de tijd dit toelaat)
      - Testen van de software voor het alternatief communicatie kanaal.
      - Schrijven van het ontwerp rapport

   c. Te gebruiken technieken
      De student zal voor het modelleren van de middleware de tekentechniek UML gebruiken.

   d. Te vermelden nadrukken

8. Risico's en maatregelen

| Risico | Kans | Impact | Maatregel impact |
|---|---|---|---|
| De student heeft geen nomadic device om op te werken | Klein | Groot | De student kan eigen nomadic device gebruiken. |
| Geen auto met een OBU om volledig te kunnen testen | Groot | Middel | Geen |
| De student is langdurig ziek | Klein | Groot | De student zal op een ander tijdstip moeten afstuderen |

9. Op te leveren (tussen)producten
De student dient de volgende producten op te leveren:

a. Vision document
b. Architectuur rapport
c. Ontwerp rapport
d. Testplan
e. Source code
f. Prototype welke een succesvolle koppeling tussen de OBU en de nomadic device bewijst

**Benodigde Beroepstaken / activiteiten**
A1 Analyseren van het probleemdomein
A3 Achterhalen van behoeften van belanghebbenden
A4 Kiezen van een ontwikkelstrategie en ontwikkelmethodiek
A5 Opstellen van systeemeisen (requirements)
C8 Ontwerpen van een technische informatie systeem
C12 Ontwerpen van een gedistribueerd systeem
C13 Het betrekken van real-time aspecten bij een ontwerp
D16 Realiseren van software
D17 Testen van software systemen
D19 Realiseren van een gedistribueerd systeem

G1 Praktische aspecten hanteren in projecten
G2 Samenwerken in project-teams
H5 Professioneel werken: Zelfstandig werken
H6 Professioneel werken: Resultaatgericht werken
H7 Professioneel werken: methodisch werken
J14 Communiceren in organisaties

**Te demonstreren beroepstaken en wijze waarop**
De student zal enkele beroepstaken moeten demonstreren. De taken die de student heeft gekozen zijn:
- A1 Analyseren van het probleemdomein. Dit gaat de student bewijzen in de elaboratiefase waar de student uitgebreid uitzoekt hoe de OBU communiceert via welke kanalen en welke protocollen er gebruikt worden, de student zal ook alle communicatie mogelijkheden van de nomadic device onder handen nemen. Hier wordt gekeken naar zowel de mogelijke kanalen tussen OBU en nomadic device als de nomadic device en de SPITS back-office.

- D16 Realiseren van software. Om deze taak te demonstreren zal de student de code voor de middleware moeten schrijven. Dit wordt aan de hand van het ontwerp gedaan.

- D17 Testen van software systemen. De student zal een uitgebreid testplan maken waar de connectie tussen nomadicdevice en OBU uitvoerig getest wordt

Alle taken die de student gaat uitvoeren worden op niveau 3 uitgevoerd.

**Persoonlijke verantwoording van keuze opdracht/bedrijf/competenties**
Logica is een bekend bedrijf binnen de informatica sector en heeft veel leuke en innovatieve opdrachten. Het bedrijf investeert tijd en geld in het begeleiden van studenten zowel op de HBO als de WO.
De opdracht is innovatief en heeft een grote kans om in de toekomst op een grote schaal het autorijden te verbeteren. De opdracht is een stap in de richting van veilig rijden.

**Nominale duur afstudeerperiode**
De opdracht duurt 17 weken.

## Activiteitenplanning

| Week | Inceptiefase | Elaboratiefase | Constructiefase 1 | Constructiefase 2 | Constructiefase 3 |
|---|---|---|---|---|---|
| 1 | • Vision document schrijven | | | | |
| 2 | • Vision document schrijven | | | | |
| 3 | | • CALM documentatie lezen<br>• Uitzoeken werking en protocol USB, BT, WIFI | | | |
| 4 | | • CALM documentatie lezen<br>• Uitzoeken werking en protocol USB, BT, WIFI | | | |
| 5 | | • Uitzoeken werking en protocol USB, BT, WIFI<br>• Eisen opstellen | | | |
| 6 | | • Eisen opstellen<br>• Advies schrijven over gekozen kanaal<br>• Schrijven architectuurrapport | | | |
| 7 | | • Schrijven architectuurrapport | | | |
| 8 | | | • Bouwen Software OBU | | |
| 9 | | | • Bouwen Software OBU | | |
| 10 | | | • Bouwen Software OBU<br>• Testen OBU software<br>• Schrijven ontwerp rapport | | |
| 11 | | | | • Bouwen software | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | nomadic device | |
| **12** | | | | • Bouwen software nomadic device | |
| **13** | | | | • Bouwen software nomadic device<br>• Testen software OBU en nomadic device<br>• Schrijven ontwerp rapport | |
| **14** | | | | | • Bouwen software alternatief kanaal |
| **15** | | | | | • Bouwen software alternatief kanaal |
| **16** | | | | | • Bouwen software alternatief kanaal<br>• Testen alternatief kanaal<br>• Schrijven ontwerp rapport |
| **17** | Uitloop week | Uitloop week | Uitloop week | Uitloop week | Uitloop week |

Naam opdrachtgever: Bert de Neef
Handtekening voor akkoord:
Datum:

Naam bedrijfsmentor: Bert de Neef en Ferry de Jong
Handtekening voor akkoord
Datum:

Naam begeleider/examinator: Hans Witkamp
Handtekening voor akkoord:
Datum:

Naam expert/examinator: Cobie van der Hoek
Handtekening voor akkoord:
Datum:


# Spits in-vehicle koppelingsprotocol voor een mobiel apparaat

| | |
|---|---|
| Document | :Plan van Aanpak |
| Auteur | :Jessie Hernandez |
| Datum | :12 mei 2010 |
| Versie | :1.0 |
| Afstudeerbedrijf | : Logica Nederland BV |
| Divisie | : |
| Competence | : |
| Bedrijfsbegeleiders | :Bert de Neef, Edwin Essenius |
| Bedrijfsmentor | :Richard Spruit |
| Onderwijsinstelling | :De Haagse Hogeschool |
| Opleiding | :Technische informatica |
| Afstudeercoördinator | :Hans Witkamp |
| Examinator | :Cobie van der Hoek |

DE HAAGSE HOGESCHOOL

WORKING TOMORROW
TO TRY IS TO LEARN

**Lokatie: Rotterdam** Lichting **10** Datum **12 mei 2010**

Logica is a business and technology service company, employing 39,000 people. It provides business consulting, systems integration and outsourcing to clients around the world, including many of Europe's largest businesses. Logica creates value for clients by successfully integrating people, business and technology. It is committed to long term collaboration, applying insight to create innovative answers to clients' business needs.

Logica is listed on both the London Stock Exchange and Euronext (Amsterdam) (LSE: LOG; Euronext: LOG).

More information is available at www.logica.com.

## Versiebeheer

| Versie | Omschrijving |
|---|---|
| v 0.1 | Conceptversie, opgeleverd ter controle door bedrijfsbegeleiders |
| V 0.2 | Samenvatting verbeterd. Hoofdstuk 2.3 uitgewerkt. Commentaar van Bert verwerkt. |
| V 0.3 | Detailplanning gemaakt, FigurenLijstbijgewerkt |
| V 0.4 | Feedback van Edwin verwerkt. |
| V1.0 | Final versie opgeleverd. |

**Tabel 9: Versiebeheer**

## Distributielijst

| Naam | Instantie | Rol / Functie |
|---|---|---|
| Richard Spruit | Logica Nederland BV | Bedrijfsmentor |
| Edwin Essenius | Logica Nederland BV | Bedrijfsmentor |
| Bert de Neef | Logica Nederland BV | Opdrachtgever |
| Hans Witkamp | De Haagse Hogeschool | Begeleider |
| Cobie van der Hoek | De Haagse Hogeschool | Expert |

**Tabel 10: Distributielijst**

# Adresgegevens

**Examinandus**

| | |
|---|---|
| Naam | :Jessie Hernandez |
| Straat / Nummer | :Stamkartplein 278 |
| Postcode / Plaats | :2521ES |
| Telefoonnummer | :0620280558 |
| E-mailadres | : |
| Onderwijsinstelling | :De Haagse Hogeschool |
| Afstudeerrichting | :Technische Informatica |
| Studentnummer | :20050537 |

**Examinator**

| | |
|---|---|
| Naam | :Hans Witkamp |
| E-mailadres | :J.W.Witkamp@hhs.nl |
| Telefoonnummer | :+31 (0)70 445 8520 |

**Bedrijfsgegevens**

| | |
|---|---|
| Naam | :Logica Nederland BV |
| Afdeling | :Working Tomorrow |
| Straat / Nummer | :George Hintzenweg 89 |
| Postcode / Plaats | :3068AX |

**Bedrijfsbegeleiders**

| | |
|---|---|
| Naam | :Edwin Essenius |
| Telefoonnummer | : +31 88 56 47717 |
| E-mailadres | :edwin.essenius@logica.com |
| Functie | :architect working tomorow |

| | |
|---|---|
| Naam | :dhr. Bert de neef |
| Telefoonnummer | :030-6027492/0622417451 |
| E-mailadres | :bert.de.neef@logica.com |
| Functie | :Interim Manager |

**Bedrijfsmentor**

| | |
|---|---|
| Naam | :Drs. Richard Spruit |
| Telefoonnummer | :+31 (0) 102537000 |
| E-mailadres | :richard.spruit@logica.com |
| Functie | :Experienced Software Engineer |

**Practice Manager**

| | |
|---|---|
| Naam | :Dr. A.M. Wesselman |
| Telefoonnummer | :+31 (0)703756000 |
| E-mailadres | :bertram.wesselman@logica.com |
| Practice | : Technical software engineering |

# Inhoud



## TabelLen en figurenlijst

## Samenvatting

Dit document bevat afspraken die gemaakt zijn met de opdrachtgever. Verder bevat dit document ook een planning met de werkzaamheden van de student gedurende deze opdracht

# Omgeving en Organisatie

## Logica

Logica is een belangrijke international speler op het gebied van IT en business services met 39.000 mensen in dienst in 36 landen. Ze bieden oplossingen en diensten aan in het domein van consultancy, design, systeem integratie , bedrijfskritische applicaties en de outsourcing van bedrijfsprocessen. Logica richt zich op vier marktsectoren – Energy Utilities en Telecom , Finance, Public Sector, en Industrie Distribution en Transport. Door deze focus zijn ze in staat om op maat gemaakte oplossingen te bieden voor de uitdagingen waar klanten in hun specifieke markt mee te maken krijgen. In Nederland is deze focus verankerd in view Industrie Sectoren, afdelingen die zich richten op een een van de segmenten.

Logica is gedreven op klanten te helpen om leidende posities te behalen en te behouden in hun indivuele marken. De kracht van Logica ligt daarbij op het gebied van industrie- en Domeinkennis, de sterke bedrijfskundig- en technologisch inzicht, en de vele successen met het leveren op tijd en binnen budget.

Het huidige Logica plc. is op 30 december 2002 ontstaan uit het voormalige Logica plc. (60%) en het voormalige CMG plc. (40%). CMG was in Nederland aanzienlijk groter dan Logica, die vooral een focus had in Groot Brittannië. In 2006 zijn daar nog WM-Data (zwaartepunt in Scandinavië) en Unilog (Zwaartepunt in Frankrijk) aan toegevoegd, zodat een gespreide dekking in Europa verkregen is. In 2008 is onder leiding van de nieuwe CEO, Andy Green, een programma opgestart om van alle onderdelen binnen het bedrijf een uniforme organisatie neer te zetten - One Logica. Hierbij zijn de waarden gedefinieerd, volgens welke iedere Logicaan zou moeten werken: Committed, Innovative en Open. De missie die hierbij gesteld is, is "To be the most trusted innovation partner".

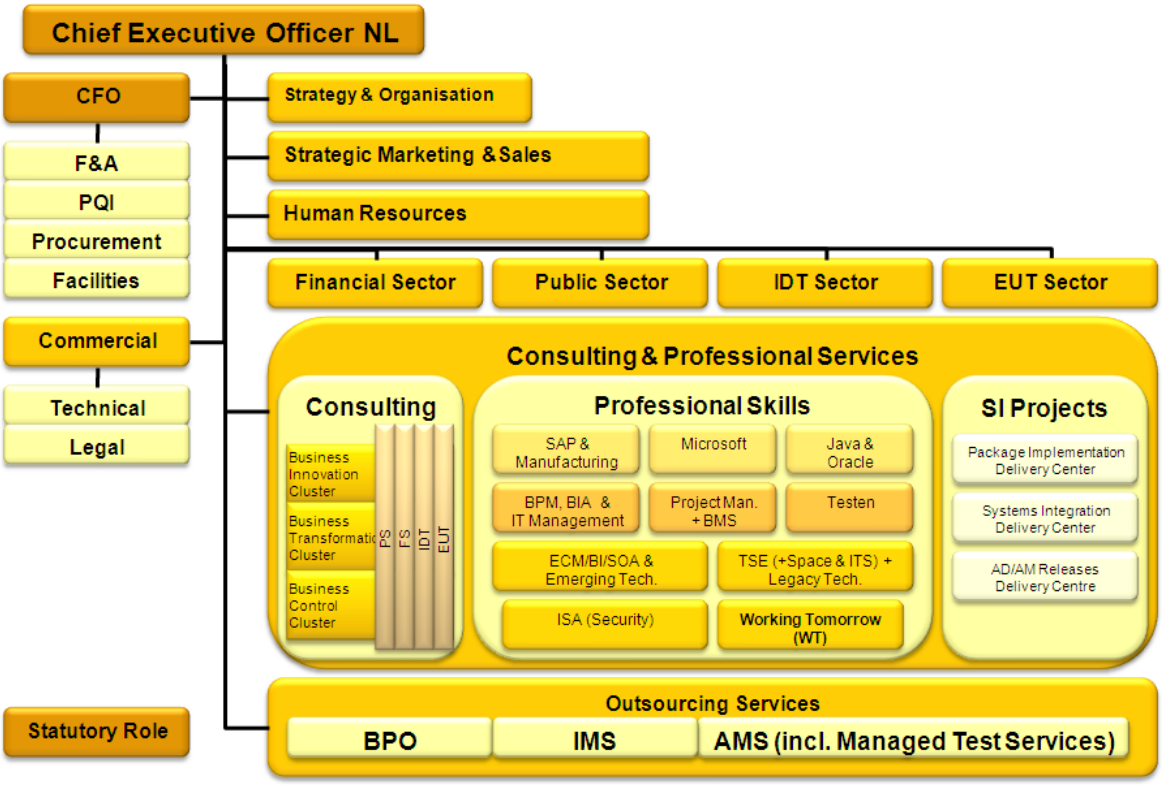
## Working Tomorrow

De opdracht wordt uitgevoerd binnen het programma Working Tomorrow. Logica heeft dit programma opgestart om studenten de gelegenheid te geven af te studeren op een innovatieve opdracht met goede begeleiding. Innovatie staat centraal. Innovatief wat betreft technologie, concept of methodiek. Zo werken er studenten aan agent technologie, Multi touch technologieën, Augmented Reality, Smart Grids en slimme meters en mobiele oplossingen. Working Tomorrow is nu op 6 vestigingen van Logica in Nederland aanwezig, en biedt jaarlijks plaats aan circa 120 studenten van HBO en WO. Op iedere vestiging is een projectleider van Working Tomorrow die de studenten begeleidt.

Het Working Tomorrow programma heeft 4 hoofddoelen:

1. Een centrale plaats bieden waar studenten uitdagende en innovatieve afstudeerprojecten kunnen uitvoeren.
2. Het werven van toekomstige werknemers.
3. Verhogen van de reputatie van Logica op het gebied van innovatie.
4. Demo's en resultaten van projecten gebruiken voor het verkrijgen van betaalde opdrachten en vergaren van kennis.

Logica Nederland heeft naast de Industrie Sectoren ook Service Lines (SL). Consulting en Professional Services (C&PS) is er op gericht om de IS zo goed mogelijk te voorzien van goed en gekwalificeerd personeel. C&PS is verder onder te verdelen in Consulting, SI Projects en Professional Skills. Professional Skills is weer verder onderverdeeld in practices, soort van afdelingen die gericht zijn op een bepaalde skillset, bijv Project Managenent, Java.Oracle of ISA.




**Figuur 36: Organogram Logica met Working Tomorrow**

Alle studenten die bij WT afstuderen zijn daarnaast ook onder gebracht in een practice die nauw aansluit bij de opdracht.

## SPITS

SPITS is een Nederlands project, belast met het creëren van 'Intelligent Traffic Systems' (ITS) concept die mobiliteit en veiligheid kunnen verbeteren. ITS maken dit allemaal mogelijk. Het idee is dat alle weggebruikers de kans krijgen om rechtstreeks met elkaar of met access points langs de weg te communiceren, en op deze manier data uit te wisselen. De uitdaging is om deze communicatie stromen samen te brengen in een groot netwerk. Cooperative vehicle infrastructure systems (CVIS), een Europees onderzoeksproject en ontwikkeling project werd gestart om deze uitdaging aan te gaan. SPITS is een Nederlands initiatief gefinancierd door een consortium van dertien partners en het Nederlandse Ministerie van Economische Zaken. Logica is een belangrijke speler in de versnelling van de concepten van CVIS in de richting van de markt in de SPITS consortium

## Begeleiding

Alle studenten die afstuderen bij Working Tomorrow, worden bij een practice geplaatst die past bij de richting het project. Een student krijgt naast procesmatige begeleiding vanuit Working Tomorrow ook inhoudelijke begeleiding vanuit de practice. Dit project valt onder de Technical Software Engineering.

# Definitie Afstudeeropdracht

## Aanleiding

Aanleiding voor dit project is de ontwikkeling binnen het SPITS project van concept producten voor derde partijen die diensten leveren voor autorijders. Voorbeelden hiervan zijn verzekeraars, leasemaatschappijen, autoverhuurbedrijven en andere mogelijke wagenparkbeheerders. SPITS wil oplossingen wegzetten waarmee dergelijke partijen diensten kunnen worden aangeboden zoals een remote-diagnostics functionaliteit (het op afstand monitoren van de toestand van een voertuig), een emergency-call functionaliteit (het automatisch inschakelen van hulpdiensten in geval van een ongeval), een stolen-vehicle functionaliteit (het tracken en tracen van gestolen auto's), of een pay-as-you-drive functionaliteit (het gebruik maken van het rijgedrag voor het bepalen van kosten zoals huurprijs, leaseprijs, of een verzekeringspremie). Hierbij willen we gebruik maken van een vast kastje in de auto: de OBU (On Board Unit). De OBU is een in-vehicle device (of mogelijk combinatie van meerdere apparaten), die deel uitmaakt van het SPITS platform en waarop SPITS applicaties kunnen draaien. Binnen SPITS wordt momenteel gewerkt aan de ontwikkeling van dergelijke in-vehicle systemen, die beschikken over standaard, vaste, communicatiekanalen, met de daarbij behorende communicatiekosten. De On Board Unit beschikt over mogelijkheden om te communiceren met de SPITS back-office, maar het is nog onduidelijk wat de kosten hiervan zullen zijn omdat de OBU nog een experimenteel product is.

## Probleemstelling

Om onafhankelijk te kunnen zijn van een specifieke Telecom leverancier, willen we de mogelijkheid hebben om ook via een ander kanaal datacommunicatie te realiseren, zonder gebruik te maken van de communicatie mogelijkheden van de OBU.

## Verankering

Tijdens dit project ga ik de focus leggen op de volgende punten. Er moet bepaald worden welke communicatie kanalen er gebruikt zullen worden. Er zijn 4 mogelijke kanalen om uit te kiezen. (Bluetooth, USB, WIFI en Infrarood). Niet alle kanalen zijn echter bruikbaar. Infrarood wordt niet overwogen, omdat dit kanaal bijna op geen enkele nomadic device [†] gebruikt wordt en dit is een "Line of sight" kanaal, dit maakt de connectie tussen OBU en Nomadic device vrijwijl onmogelijk. De kanalen die overblijven (Bluetooth, USB en WIFI) worden wel overwogen om gebruikt te worden als communicatie kanaal tussen OBU en Nomadic device

## Relevantie

Deze opdracht wordt uitgevoerd om uit te zoeken hoe er communicatie tussen een OBU en een nomadic device verwezenlijkt kan worden. Deze communicatie is nodig om de connectiviteit van de OBU flexibel te maken. Dit kan de data transmissie kosten van de OBU aanzienlijk verminderen

[†] Een mobiel apparaat met een Bluetooth, USB of WIFI communicatie kanaal

## Antwoord

Het realiseren van een proof of concept van een middlewarelaag om datacommunicatie van de OBU met de SPITS back-office tot stand te brengen via een nomadic device. De volgende producten worden opgeleverd.

### *Vision Document*
Dit document. Het Vision document bevat de afspraken met de opdrachtgever. In dit document kan je de aanpak van de te gebruiken technieken en de planning terugvinden.

### *Architectuur rapport*
Dit rapport zal worden opgesteld in de elaboratie fase en zal een advies bevatten met het aanbevolen communicatie kanaal. Doordat uiteindelijk twee kanalen gebouwd zullen worden zal in het advies ook naar voren komen welk kanaal prioriteit heeft boven de andere.

### *Ontwerp rapport*
Het ontwerp rapport zal het ontwerp van de software bevatten. Deze wordt geschreven in de constructie fase. Dit zal een zeer technisch document worden met het ontwerp gemodelleerd in UML.

### *Mastertestplan*
Het mastertestplan bevat een testplan die alle mogelijke scenario's beschrijft en de bijbehorende verwachtingen. Met het mastertestplan kan worden gecontroleerd of de applicatie aan de eisen voldoet.

### *Broncode*
de gehele broncode van de software wat geschreven zal worden.

### *Prototype*
De prototype is een demonstratieapplicatie waarin een succesvolle koppeling tussen de OBU en een Android toestel bewijst. Deze wordt opgeleverd op een cd-rom met een bijgevoegde handleiding. Deze applicatie wordt dan ook uitvoerig getest in de Mastertestplan.

## Strategie

Om dit project overzichtelijk te maken heb ik gekozen om met het software ontwikkel proces RUP te werken, om de software die ik ga ontwerpen ga ik UML gebruiken.

### *RUP*

RUP staat voor Rational Unified Process, een systeemontwikkelingmethodiek. Binnen deze methodiek wordt de klemtoon gelegd op risico's en softwarearchitectuur.
Tijdens het project worden alle fases van RUP doorgenomen. Niet alle fases worden uitgevoerd zoals die beschreven zijn met name de Transitie fase. Deze wordt meer gezien als overdracht van het project en niet als uitrol fase zoals beschreven in RUP.

### *UML*

De modelleertaal UML is inmiddels uitgegroeid tot de wereldwijde standaard voor het modelleren van eisen, functionaliteit, componenten en services. UML kent daartoe een groot aantal modelleertechnieken zoals use case diagrammen, activiteiten diagrammen, sequentie diagrammen, communicatie diagrammen, klassen diagrammen en package diagrammen. Deze modelleertechnieken worden gebruikt tijdens de verschillende fasen van systeemontwikkelprojecten, zoals analyse, ontwerp, bouw en zelfs tijdens testen. De specificaties van UML definiëren echter vooral de structuur van deze modelleertechnieken en geven maar zeer beperkt aan hoe ze in de praktijk het best zijn toe te passen.
Van UML gebruik ik de ontwerptechnieken om Use Cases, klassendiagram, en sequentie diagrammen op te stellen.

## Randvoorwaarden

De software wat geschreven wordt, wordt op speciale hardware geschreven daarvoor is deze hardware noodzakelijk. Het gaat om de volgende

*hardware:*

- een OBU
- een nomadic device met het Android besturingssysteem
- toegang tot de SPITS back-office

*Software*

Er is software nodig om het project succesvol te laten verlopen

- De software development kit van het Android besturing systeem

## Risico's

| Risico | Kans | Gevolg | Maatregel |
|---|---|---|---|
| Ik heb geen nomadic device ter mijn beschikking | Klein | Groot | Ik kan mijn eigen nomadic device gebruiken |
| Ik heb geen OBU ter mijn beschikking | Klein | Groot | Het project gaat niet meer door. De OBU is een randvoorwaarde |
| Ik heb geen auto met OBU om volledig invloeden van buiten te testen | Groot | Middel | Geen (er wordt dan niet getest of er invloeden van buiten zijn bij de connectie tussen nomadic device en OBU) |
| Ik ben langdurig ziek | Klein | Groot | Met de Hogeschool kan uitstel geregeld worden |
| Data verlies | Klein | Groot | Elk document wordt op de netwerkschijf van Logica geplaatst. Als de gegeven ruimte vol is zal deze op een persoonlijke hardeschijf/flashdrive worden gedaan |

**Tabel 11: Lijst met Risico's**

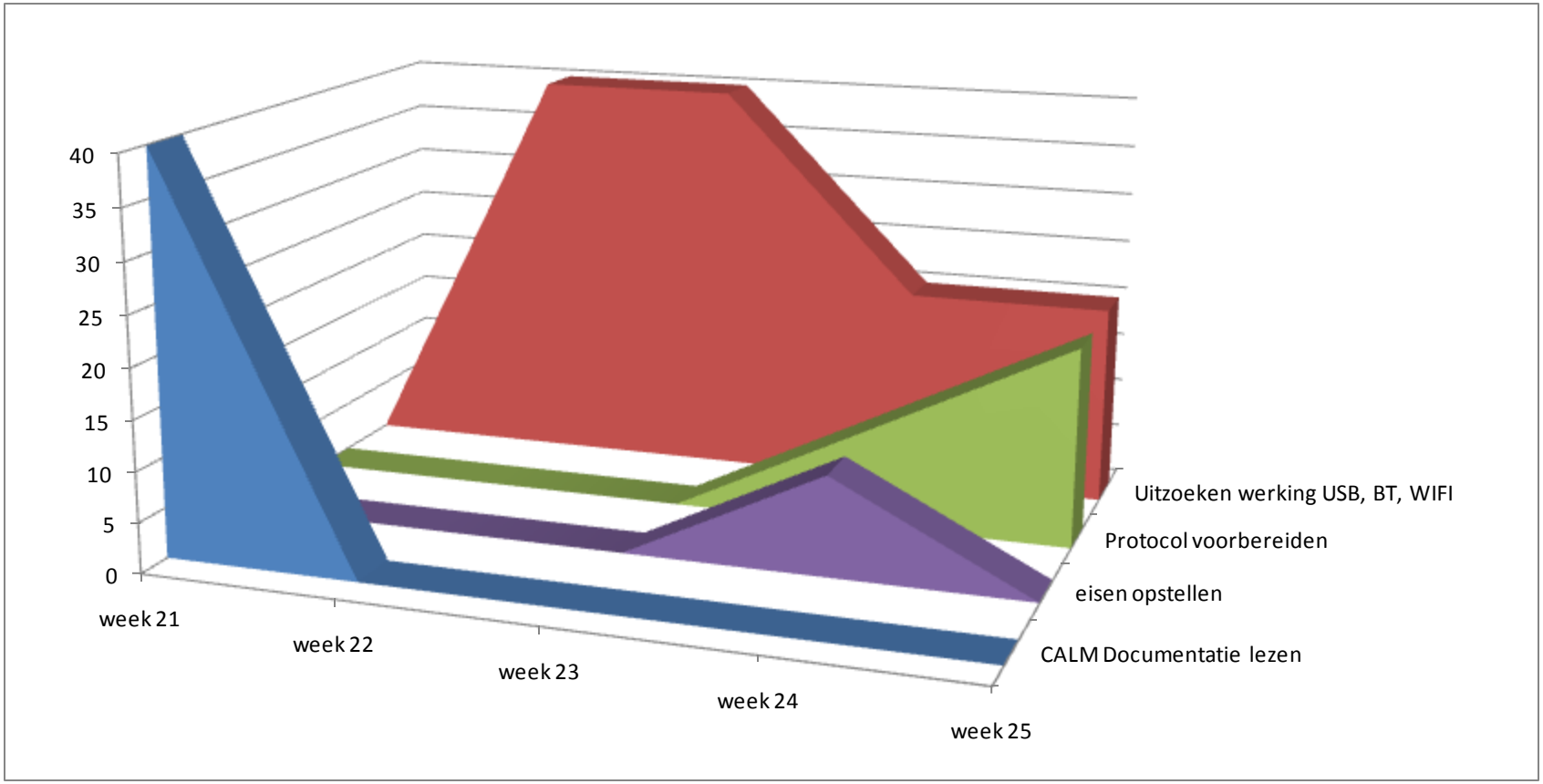
# Planning

## Mijlpalen

| Mijlpaal | Deadline |
|---|---|
| PvA wordt opgelevered | 28 mei 2010 |
| Architectuur rapport | 25 juni 2010 |
| Constructie rapport (iteratie 1) | 16 july 2010 |
| Constructie rapport (iteratie 2) | 6 augustus 2010 |
| Constructie rapport (iteratie 3) | 27 augustus 2010 |
| Master testplan | 30 juli 2010 |
| Mastertestplan uitgevoerd | 3 september 2010 |
| Concept afstudeer verslag (1$^{e}$ opzet) | 2 juli 2010 |
| Concept afstudeer verslag (2$^{e}$ opzet) | 30 juli 2010 |
| Concept afstudeer verslag (3$^{e}$ opzet) | 15 augustus 2010 |
| Oplevering afstudeer verslag | 23 september 2010 |

**Tabel 12: Mijlpalen**

## Iteratie planning




**Figuur 37: Detail planning Elaboratiefase**

## Begrippenlijst

| Afkorting | Betekenis |
|---|---|
| OBU | On Board Unit |
| SPITS | Strategic Platform For Intelligent Traffic Systems |
| CVIS | Cooperative vehicle infrastructure systems |
| RUP | Rational Unified Process |
| UML | Unified Modeling Language |
| Nomadic Device | Een mobiel apparaat met een Bluetooth, USB of WIFI communicatie kanaal |

**Tabel 13: Begrippenlijst**


# Spits in-vehicle koppelingsprotocol voor een mobiel apparaat

| | |
|---|---|
| Document | Architectuur Rapport |
| Auteur | :Jessie Hernandez |
| Datum | :4 juni 2010 |
| Versie | :0.3 |
| Afstudeerbedrijf | : Logica Nederland BV |
| Divisie | : |
| Competence | : |
| Bedrijfsbegeleiders | :Bert de Neef, Edwin Essenius |
| Bedrijfsmentor | :Richard Spruit |
| Onderwijsinstelling | :De Haagse Hogeschool |
| Opleiding | :Technische informatica |
| Afstudeercoördinator | :Hans Witkamp |
| Examinator | :Cobie van der Hoek |

DE HAAGSE HOGESCHOOL

WORKING TOMORROW
TO TRY IS TO LEARN

Logica is a business and technology service company, employing 39,000 people. It provides business consulting, systems integration and outsourcing to clients around the world, including many of Europe's largest businesses. Logica creates value for clients by successfully integrating people, business and technology. It is committed to long term collaboration, applying insight to create innovative answers to clients' business needs.

Logica is listed on both the London Stock Exchange and Euronext (Amsterdam) (LSE: LOG; Euronext: LOG).

More information is available at www.logica.com.

**Versiebeheer**

| Versie | Omschrijving |
|---|---|
| v 0.1 | Conceptversie gemaakt |
| V0.2 | Feedback van richard verwerkt (huidige situatie, Eisen bijgesteld, Use cases bijgesteld) |
| V0.3 | Use Cases met Edwin besproken en uitgeschreven. Klassendiagrammen verbeterd. |

**Tabel 14: Versiebeheer**

**Distributielijst**

| Naam | Instantie | Rol / Functie |
|---|---|---|
| Richard Spruit | Logica Nederland BV | Bedrijfsmentor (SPITS) |
| Edwin Essenius | Logica Nederland BV | Bedrijfsmentor (WT) |
| Bert de Neef | Logica Nederland BV | Opdrachtgever |
| Hans Witkamp | De Haagse Hogeschool | Begeleider |
| Cobie van der Hoek | De Haagse Hogeschool | Expert |

**Tabel 15: Distributielijst**

# Inhoud

# Figurenlijst

# Samenvatting

SPITS is een Nederlands project, belast met het creëren van Intelligent Traffic Systems (ITS) concepten die mobiliteit en veiligheid kunnen verbeteren. Spits is een vervolg project op het CVIS project. CVIS is een Europees onderzoek en ontwikkel project met als doel het ontwikkelen, bouwen en testen van de technologie die er nodig bij de communicatie tussen auto's onderling en met het infrastructuur langs de weg.

In dit rapport wordt het architectuur van de communicatie link tussen ATOP en nomadic device beschreven om de communicatie vanuit de ATOP met de BackOffice te maken. In hoofdstuk 1 wordt de huidige situatie uitgelegd met wat de bedoeling van het project is. Hoofdstuk 2 zijn de eisen uitgewerkt deze zijn opgesplitst in funtionele en niet functionele eisen. Hoofdstuk 3 zijn de use cases beschreven. In dit hoofdstuk komt ook een use case diagram voor. In hoofdstuk 4 zijn de concept klassendiagrammen beschreven. De middleware zal 2 klassendiagrammen hebben omdat de middleware op de ATOP en nomadic device geïmplementeerd wordt. In hoofdstuk 5 wordt een gereviseerde lijst met risico's behandeld. Deze risico's zijn pas in een later stadium aan het licht gekomen. Hoofdstuk 6 komend e mijlpalen voor de elaboratiefase aan de orde. In het laatste hoofdstuk, hoofdstuk 7 komt de planning voor de constructiefase voor.

## Huidige situatie

Binnen het SPITS architectuur zijn er verschillende applicaties met verschillende communicatie behoeften. Zo kun je de applicaties in drie groepen delen.

6. Applicaties die gebruik maken van een connectie tussen de auto's onderling
7. Applicaties die gebruik maken van een connectie tussen auto en weg infrastructuur
8. Applicaties die gebruik maken van een connectie tussen auto en BackOffice

In dit project wordt de connectie tussen auto en BackOffice ontworpen en ontwikkeld. Voor deze connectie is er een "On Board Unit" (ATOP) nodig. Deze ATOP heeft verschillende communicatie kanalen elke met zijn eigen voor- en nadelen. De communicatie kanalen zijn:

- GSM
- UMTS
- GPS
- USB
- WIFI

Voor een connectie vanuit de auto naar de Back office is niet elk kanaal geschikt. de mogelijke communicatiekanalen die direct inzetbaar zijn de GSM en UMTS. Voor deze communicatiekanalen zijn echter SIM-kaarten nodig die geleverd worden door Telecom providers. Aan de service die de Telecom Provider bied zijn kosten gebonden. Deze kosten kunnen erg hoog oplopen, om deze kosten te verlagen wordt er naar andere communicatie kanalen op de ATOP gekeken.

Als een additionele communicatie kanaal is een communicatie link tussen de USB kanaal van de ATOP met een nomadic device die een mobiele internet connectie heeft. Op deze manier zou je de nomadic device kunnen gebruiken om de BackOffice te benaderen.

# Eisen

In dit hoofdstuk worden de eisen voor de middleware beschreven. De verschillende eisen zijn opgesplitst in functionele en niet functionele eisen.

## Functionele eisen

- Middleware moet de data naar de BackOffice sturen.
- Bij uitval van de internet link (tussen de Nomadic Device en BackOffice) moet er een melding naar de applicatie gestuurd worden.
- Als de connectie tussen nomadic device en BackOffice uitvalt moet deze automatisch opnieuw geinitialiseerd worden.

## Niet functionele eisen

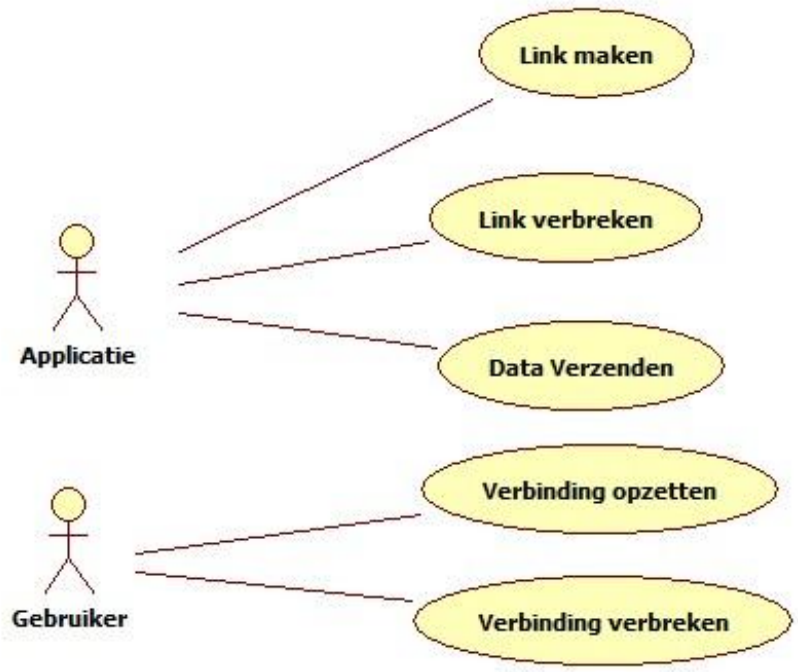
- De link tussen ATOP en nomadic device moet via USB opgezet worden
- De data moet zowel IPv4 als IPv6 ondersteunen.
- De link moet automatisch opgezet worden zonder input van de gebruiker.
- De applicatie stuurt de data synchroon, er moet altijd een response vanuit de ATOP naar de applicatie.
- De middleware draait voorlopig alleen op een Android toestel met name een HTC Hero met Android versie 1.5

# Use case

In dit hoofdstuk worden de Use cases die in de middleware gebruikt worden uitgelegd. De use cases zijn belangrijk om de eisen van de middleware in kaart te brengen.

## Use case Diagram

Hieronder wordt het use case diagram weergegeven. Dit use case diagram heeft de actortoren Gebruiker en Applicatie. Dit diagram is voort gekomen uit de eisen en wensen die in hoofdstuk 2 zijn beschreven.




**Figuur 38: Use case Diagram**

## Use Case beschrijving

Hier onder staan alle Use case beschrijvingen voor de middleware.

| **Naam** | **Link maken** |
|---|---|
| **Actor** | Applicatie |
| **Pre-Condities** | Nomadic device is aan; Nomadic device is in bereik van de ATOP; De middleware is op de nomadic device aan het draaien. |
| **Post-Condties** | Er is een connectie gemaakt tussen ATOP en nomadic device via USB en tussen nomadic device en BackOffice |
| **Flow of events** | 4. De applicatie op de ATOP wilt data gaan verzenden en meldt dit aan de ATOP.<br>5. De ATOP maakt een connectie met de nomadic device en instrueert de nomadic device om een connectie met de BackOffice te maken.[NO_USB-DEVICE_CONNECTED][BACKOFFICE NOT REACHABLE]<br>6. De applicatie gaat door met use case Data Inpakken |
| **Exepties** | [NO_USB-DEVICE_CONNECTED] Er is geen USB apparaat aan de ATOP gebonden.<br>[BACKOFFICE NOT REACHABLE] de BackOffice is niet bereikbaar. |

| **Naam** | **Link verbreken** |
|---|---|
| **Actor** | Applicatie |
| **Pre-Condities** | Er is een bestaande connectie tussen nomadic device en ATOP en tussen nomadic device en BackOffice; |
| **Post-Condties** | De link tussen nomadic device en ATOP en tussen nomadic device en BackOffice is verbroken |
| **Flow of events** | 4. De applicatie is klaar met data verzenden en laat dit aan de ATOP weten<br>5. De ATOP stuurt een bericht via de nomadic device naar de BackOffice dat de connectie verbroken moet worden.<br>6. De BackOffice beëindigt de connectie met de nomadic device. |
| **Exepties** | GEEN |

| **Naam** | **Data verzenden** |
|---|---|
| **Actor** | Applicatie |
| **Pre-Condities** | Er is een bestaande connectie tussen nomadic device en ATOP en tussen nomadic device en BackOffice; De data is ingepakt en is verzendklaar; |
| **Post-Condties** | De data is verzonden. |
| **Flow of events** | 3. De ATOP verzend de data via de nomadic device naar de BackOffice en meldt de applicatie dat de data verzonden is.[DATA_NOT_ARRIVED]<br>4. De applicatie maakt het volgende pakket klaar om naar de ATOP verzonden te worden om te worden ingepakken. |
| **Exepties** | [DATA_NOT_ARRIVED] De data is verzonden mar is niet aangekomen. De ATOP zal proberen weer een link te leggen met de BackOffice. Zie use case Link maken. |

| Naam | **Verbinding opzetten** |
|---|---|
| **Actor** | Gebruiker |
| **Pre-Condities** | Nomadic device is aan; Nomadic device is in bereik van de ATOP; |
| **Post-Condties** | De nomadic device is klaar om een connectie met de ATOP aan te gaan. |
| **Flow of events** | 6. De gebruiker start de middleware op de nomadic device<br>7. De nomadic device kijkt in de config file naar de instellingen voor de connectie.[NO_CONFIG]<br>8. A) De nomadic device geeft de config scherm weer<br>B) De nomadic device laadt de config file en meldt de ATOP dat ie klaar met opstarten is.<br>9. A) De gebruiker vult de configuratie voor de connectie in<br>10. A) De nomadic device laadt de configuratie in en meldt de ATOP dat ie klaar met het opstarten is. |
| **Exepties** | [NO_CONFIG] Er is geen config file de nomadic device zal dan de config scherm weergeven en deze opslaan als die ingevuld is. |

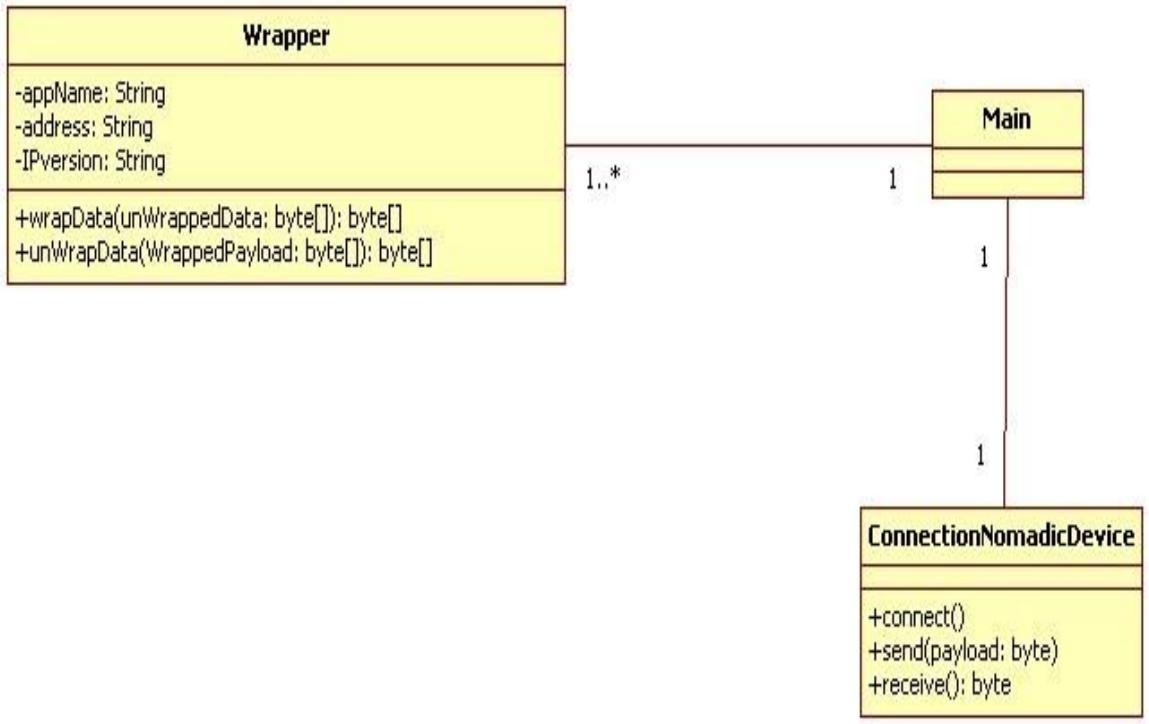
| Naam | **Verbinding verbreken** |
|---|---|
| **Actor** | Gebruiker |
| **Pre-Condities** | Nomadic device is aan; Nomadic device is in bereik van de ATOP; Er is een opgezette verbinding. |
| **Post-Condties** | De nomadic device heeft geen verbinding met de ATOP nog de Backoffice. |
| **Flow of events** | 3. De gebruiker geeft aan dat hij de connectie verbinding wilt verbreken.<br>4. De nomadic device verbreekt de verbinding. |
| **Exepties** | GEEN |

# Klassendiagram

Dit hoofdstuk bevat de verschillende klassendiagrammen voor de applicatie. Omdat de applicatie op twee delen heeft zijn er dan ook twee klassendiagrammen gemaakt. Op deze manier kan de onderscheid van de programma's duidelijk worden.

## Klassendiagram ATOP

Onderstaand is de klassendiagram voor de applicatie op de ATOP weer gegeven. In de volgende paragraaf wordt deze uitgelegd.




**Figuur 39: klassendiagram ATOP**

## Klassendiagram beschrijving

In deze paragraaf worden de klassen uit het klassen diagram voor de applicatie op de ATOP uitgelegd

## Main

De Main klasse is de hoofdklasse van het programma. In deze klasse wordt de data ingepakt en klaargemaakt om verzonden te worden. Ook in deze klasse wordt de connectie met de nomadic device geïnitieerd.

## Wrapper

Deze klasse moet de data vanuit de applicatie op de applicatie laag van de ATOP gaan verwerken. Er moeten meerdere Wrappers mogelijk zijn zodat meerdere applicaties tegelijkertijd data kunnen verzenden en ontvangen.

**Attributen**

- appname wordt gebruikt om bij te houden welke applicaties allemaal data aan het verzenden zijn naar de BackOffice. Dit attribuut is een String
- address stelt het BackOffice adres voor en wordt gebruikt om de data in te pakken
- IPversion is een weergave van de te gebruiken IP protocol bij het opzetten van de connectie link met de BackOffice.

**Methodes**

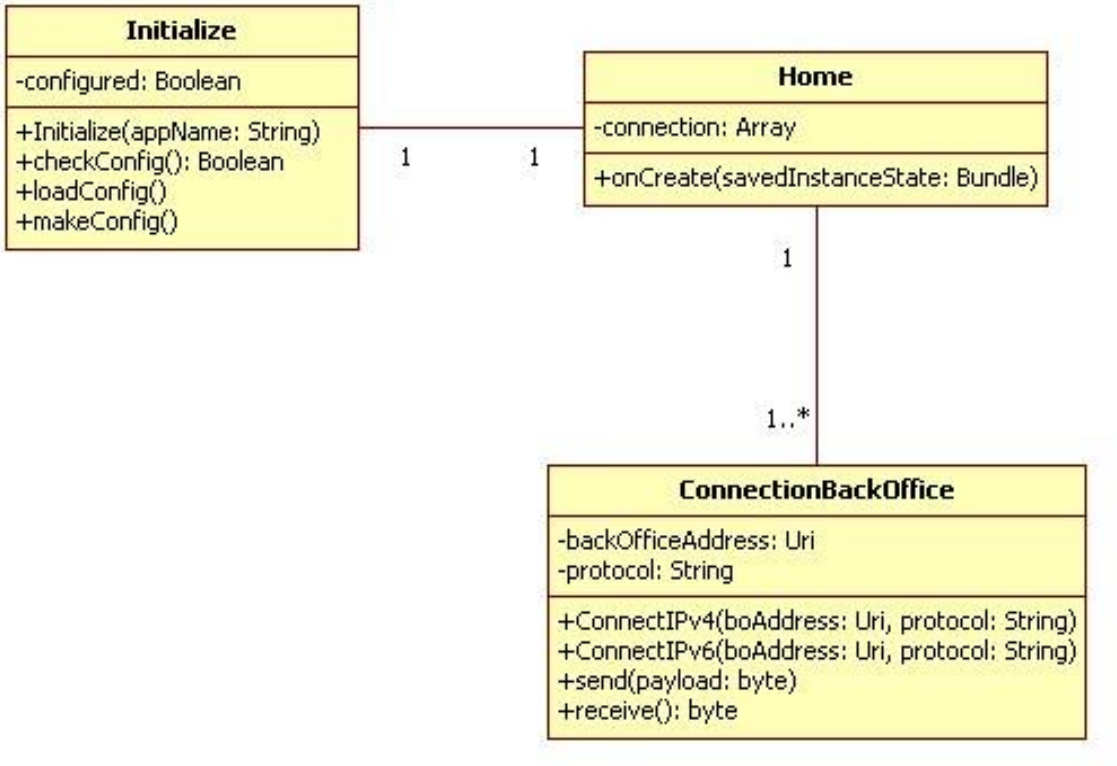
- wrapData wordt gebruikt om de data in te pakken en verstuur klaar te maken. De methode verwacht een byte array bij als hij wordt aangeroepen en nadat hij het pakket heeft ingepakken wordt de data als byte array terug gestuurd.
- unWrapData is de methode dat wordt aangeroepen wanneer data binnenkomt vanuit de nomadic device. Uit dit pakket wordt de appName gehaald om op deze manier te weten te komen voor welke applicatie het pakket bestemd is. Deze methode verwacht het pakket wat is ontvangen als een byte array en na het uitpakken wordt de data als byte array weer terug gestuurd.

## ConnectionNomadicDevice

Deze klasse zal de connectie met de Nomadic Device gaan implementeren. Deze klasse heeft de methodes connect(),send() en receive(). Zoals de methode namen het al zeggen worden deze methodes gebruikt om data te versturen en ontvangen. Ook van deze klasse moeten er meerder instanties mogelijk zijn omdat niet elke applicatie heeft dezelfde BackOffice.

## Klassendiagram Nomadic device

Onderstaand is de klassendiagram voor de applicatie op de nomadic device weer gegeven. In de volgende paragraaf wordt deze uitgelegd.




**Figuur 40: Klassendiagram Nomadic device**

## Klassendiagram beschrijving

In deze paragraaf worden de klassen uit het klassen diagram voor de applicatie op een nomadic device uitgelegd.

### Home

De Home klasse is de hoofdklasse van het programma. In deze klasse wordt de connectie met de ATOP geïnitieerd.

**Attributen**

- connection is een attribuut dat de verschillende connecties per Applicatie bijhoudt. Dit attribuut is van de type Array

## Initialize

Deze klasse zal ervoor zorgen dat de configuratie voor de connectie ingeladen wordt. Deze data wordt eenmalig aangemaakt en opgeslagen in een XML bestand. Als er geen config bestand te vinden is zal de klasse een configuratie scherm tonen.

**Attributen**

- configured is een attribuut met de type Boolean. Deze zal TRUE zijn als een configuratie file bestaat en FALSE als deze niet bestaat

**Methodes**

- checkConfig zoekt naar de config file. Als deze niet bestaat zal hij een boolean met de waarde FALSE terug sturen en waneer wel een config file gevonden is zal hij een boolean waarde TRUE terug sturen
- methode loadConfig zal de configuratie inladen.
- makeConfig zal zorgen voor het tonen van de configuratie scherm en het opslaan van de config file

## ConnectionBackOffice

Deze klasse zal de connectie met de ATOP gaan implementeren. Deze klasse heeft de methodes connect(),send() en receive(). Zoals de methode namen het al zeggen worden deze methodes gebruikt om data te versturen en ontvangen.

**Attributen**

- De backOfficeAddress attribuut wordt gebruikt om het adres van de BackOffice op te geven. Dit attribuut is van het type java.net.URI
- protocol wordt gebruikt om de protocol van de connectie op te geven. Dit is nodig om de beslissing te nemen of er een IPv6 tunnel nodig is bij het benaderen van de BackOffice. Dit attribuut is van de type String.

**Methodes**

- connectIPv4 deze methode wordt gebruikt om een connectie op te zetten met een BackOffice dat in een IPv4 netwerk zit.
- connectIPv6 deze methode wordt gebruikt om een connectie op te zetten met een BackOffice dat in een IPv6 netwerk zit
- De send methode zorgt ervoor dat de data verzonden wordt. Deze methode verwacht dat bij aanroep een byte array wordt meegestuurd. Deze byte array is de data in bytes weergegeven dat verzonden moet worden.
- De receive methode luistert naar inkomende data pakketen Deze methode zal nadat hij een pakket ontvangen heeft het pakket als byte array terug sturen

## Risico's

| Risico | Kans | Gevolg | Maatregel |
|---|---|---|---|
| nomadic device ondersteund geen USB en/of Bluetooth | Klein | Groot | Er moet dan een onderzoek gedaan worden om te kijken waar het probleem ligt. Er kan dan gekozen worden voor een nieuw toestel. |
| ATOP ondersteunt geen Bluetooth | Groot | Groot | De communicatie link tussen ATOP en nomadic device wordt alleen over USB gemaakt. |
| ATOP ondersteunt geen USB | Klein | Groot | Moet er naar een alternatief communicatie kanaal gezocht worden die de ATOP en de nomadic devices wel ondersteunen. |

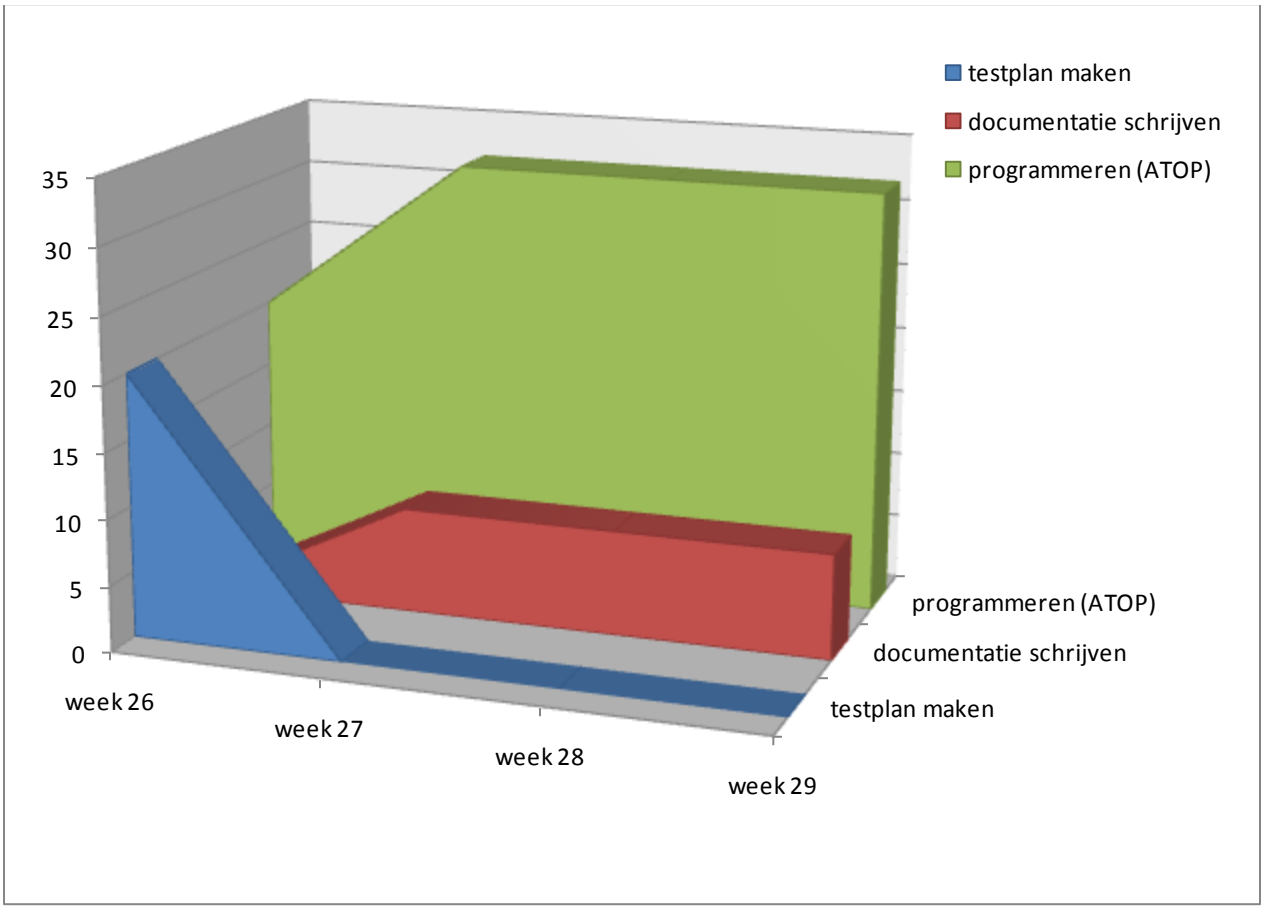
## Mijlpalen elaboratiefase

Om deze fase te kunnen afronden moeten er aan het eind bepaalde mijlpalen behaald worden. Deze mijlpalen worden in dit hoofdstuk opgesomd.

- Use cases voor minimaal 80% klaar
- De functionele en niet functionele eisen zijn in kaart gebracht
- Er is een architectuur gemaakt
- Herziene lijst van risico's
- Een planning voor de volgende fase

# Planning

In het volgend overzicht wordt weergegeven wat de planning voor de volgende fase van dit project is. In de volgende fase wordt een testplan gemaakt en wordt er een gedeelte van de middleware geschreven. Met name de USBlink op de nomadic device.




**Figuur 41: planning iteratie constructiefase 1**

# Begrippenlijst

| | |
|---|---|
| ATOP | Automotive Telematics On-board unit Platform |
| BackOffice | server dat spits applicaties host |
| GPS | Global Positioning System |
| GSM | Global System for Mobile Communications |
| Nomadic Device | Smartphone met mobiele internet connectie |
| UMTS | Universal Mobile Telecommunications System |
| USB | Universal Serial Bus |
| WIFI | Certificering handelsmerk op basis van 802.11 producten |


# Spits in-vehicle koppelingsprotocol voor een mobiel apparaat

| | |
|---|---|
| Document | Ontwerprapport Iteratie 1 |
| Auteur | :Jessie Hernandez |
| Datum | 14 september 2010 |
| Versie | 1 |
| Afstudeerbedrijf | : Logica Nederland BV |
| Divisie | :Technical Software Engineering |
| Competence | : |
| Bedrijfsbegeleiders | :Bert de Neef, Edwin Essenius |
| Bedrijfsmentor | :Richard Spruit |
| Onderwijsinstelling | :De Haagse Hogeschool |
| Opleiding | :Technische informatica |
| Afstudeercoördinator | :Hans Witkamp |
| Examinator | :Cobie van der Hoek |

DE HAAGSE HOGESCHOOL

WORKING TOMORROW
TO TRY IS TO LEARN

Logica is a business and technology service company, employing 39,000 people. It provides business consulting, systems integration and outsourcing to clients around the world, including many of Europe's largest businesses. Logica creates value for clients by successfully integrating people, business and technology. It is committed to long term collaboration, applying insight to create innovative answers to clients' business needs.

Logica is listed on both the London Stock Exchange and Euronext (Amsterdam) (LSE: LOG; Euronext: LOG).

More information is available at www.logica.com.

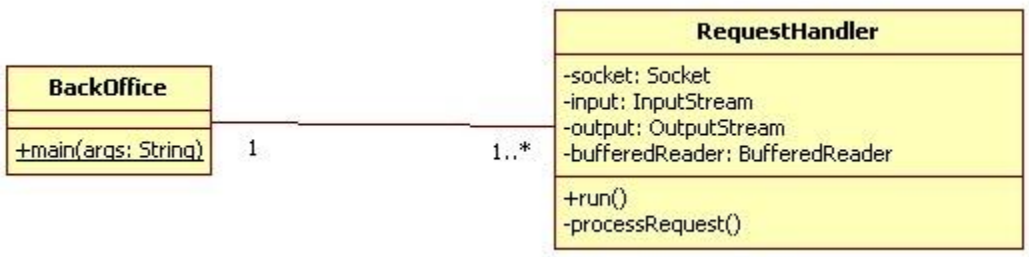
# Inleiding

In dit document wordt het ontwerp in detail besproken. Dit document is gebaseerd op de constructie fase iteratie 1. In deze fase wordt de communicatie tussen de BackOffice en de nomadic device gemoduleerd.

# BackOffice

## klassendiagram

In dit hoofdstuk wordt het klassendiagram en het sequentiediagram van de BackOffice uitgelegd.




**Figuur 42: klassendiagram BackOffice Iteratie 1**

### BackOffice

Dit is de Main klasse. In deze klasse wordt de BackOffice geinitialiseerd.

### RequestHandler

De RequestHandler is de klasse die alle verzoeken die binnen komen verwerkt. In deze klasse worden de pakketen ontvangen door middel van de BufferedReader en de InputStream. Deze klasse heeft 4 attributen en 2 methodes.

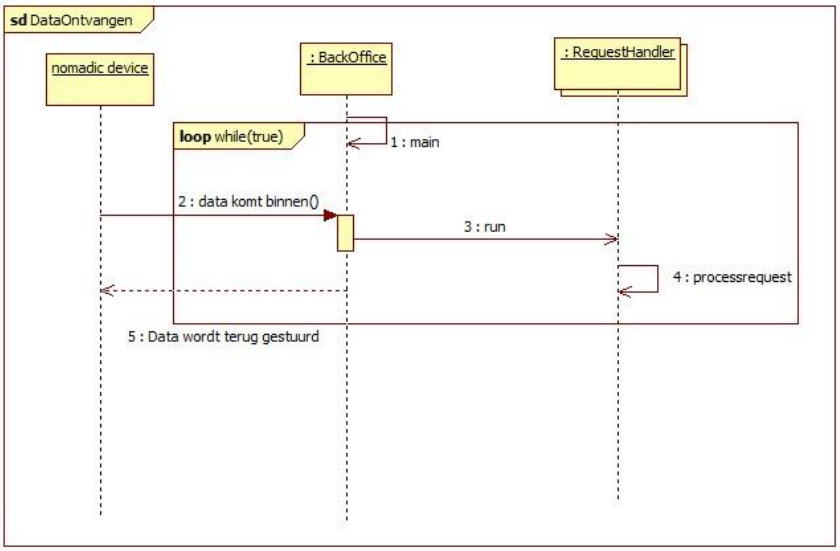
### Attributen

- Om een connectie op te zetten wordt een Socket gebruikt, een socket is een eindpunt om twee computers te laten communiceren.
- Het attribuut input van het type InputStream ontvangt alle data die via de socket binnenkomt.
- output van het type OutputStream zorgt ervoor dat de data verzonden kan worden via het netwerk.
- bufferedReader is het attribuut van het type BufferedReader dat alle binnegekomen data wordt gebuffert om in een keer wordt opgehaald in plaats van elke byte 1 read methode.

### Methode

- run: deze methode is een overschreven methode uit de Runnable interface. Deze methode wordt automatisch aangeroepen bij het aanmaken van een thread deze methode zalk de prive methode processRequest aanroepen
- processRequest is de methode die ervoor zorgt dat de data ontvangen en verwerkt wordt.

### Sequentiediagram




**Figuur 43: sequentie diagram BackOffice iteratie 1**

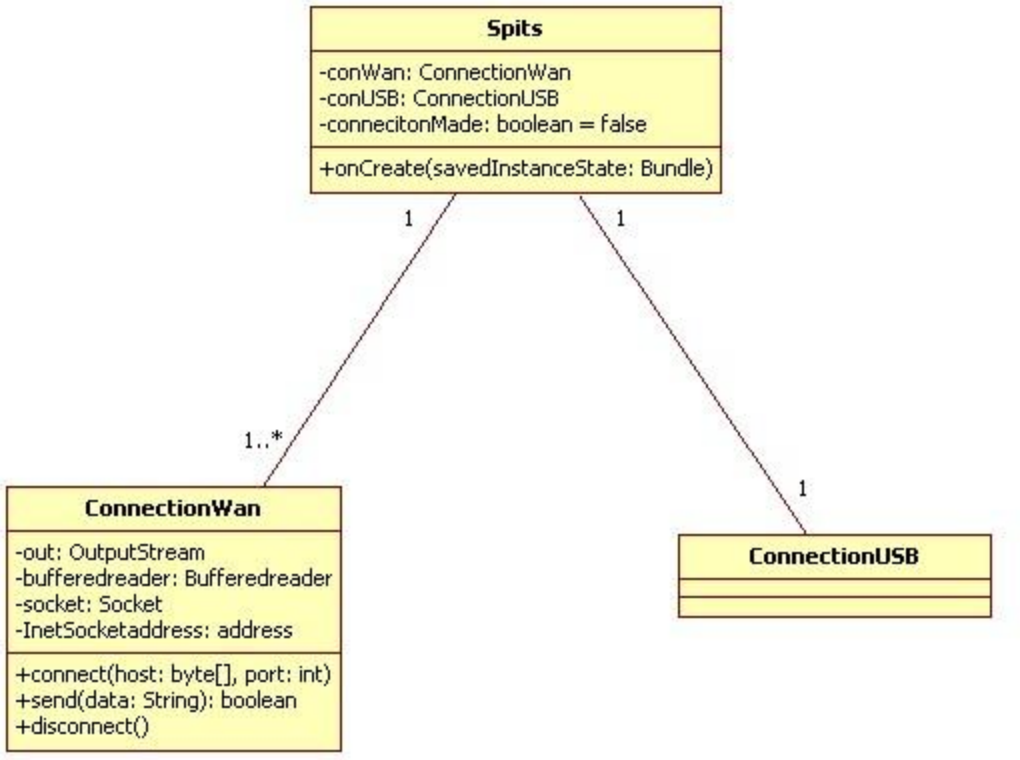
In dit diagram maakt de klasse BackOffice de BackOffice aan. De BackOffice klasse doet dit aan de hand van een ServerSocket. Wanneer deze ServerSocket is aangemaakt zal de applicatie in een oneindige lus wachten tot er data binnen komt die voor de BackOffice bestemd is. Wanneer de data binnenkomt zal de main methode de RequestHandler aanroepen en deze in een thread stoppen om op die manier meerdere connecties aan te gaan. Wanneer de data binnen is gekomen zal deze methode ook een bevestiging terugsturen.

## Android toestel

### Klassendiagram

Bij het bepalen van het klassendiagram voor de nomadic device moet ik rekening houden met twee klassen die connecties afhandelen. Maar doordat beide klassen verschillende eigenschappen hebben kies ik ervoor om geen gebruik te maken van overerving.

Ik heb verder een derde klasse, de main klasse waar de data vanuit de klasse komt waar de USB connectie geregeld wordt en door verstuurd wordt naar de klasse die de data naar de BackOffice verstuurd. In de afbeelding hieronder ziet u de klassendiagram voor de nomadic device.




**Figuur 44: klassendiagram Android toestel Iteratie 1**

In het volgende onderdeel worden de klassen van het klassendiagram beschreven.

**Spits**

Bij initialisatie van de applicatie op het Android platform zal eerst de Spits klasse worden aangeroepen. Deze klasse is een overerving van de klasse Activity. Deze klasse zal eerst een connectie met de ATOP via USB maken. Pas wanneer data binnen komt en klaar is voor het verzenden wordt de connectie met de BackOffice gemaakt.

**ConnectionWan**

De ConnectionWan klasse zorgt voor de communicatie tussen nomadic device en BackOffice. Deze klasse heeft vier attributen en drie methodes. Deze worden hieronder uitgelegd.

**Attributen**

- Het attribuut out van het type OutputStream wordt gebruikt om de data die vanuit de BackOffice komt te ontvangen.
- bufferedReader van het type BufferedReader is het attribuut die alle data ontvangt en buffert.
- socket van het type Socket wordt gebruikt om de connectie op te zetten
- address van het type InetSocket wordt gebruikt om het IP van de BackOffice op te geven.

**Methodes**

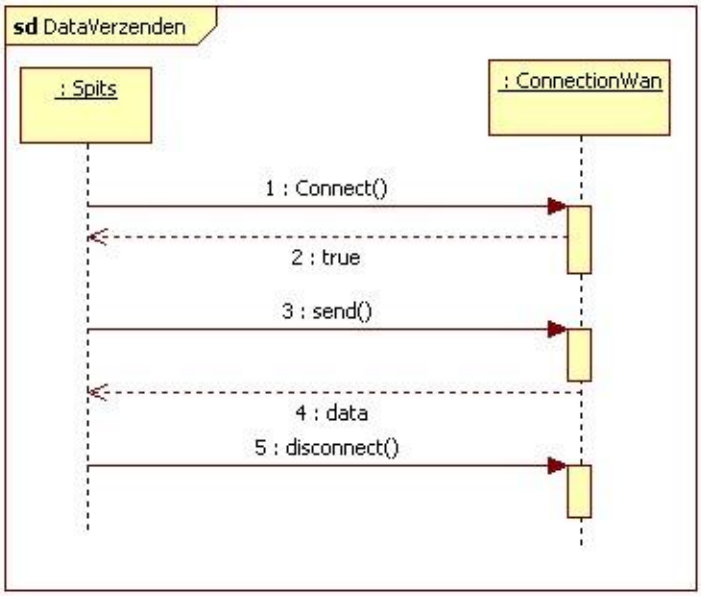
- De methode connect () zorgt voor het initialiseren van de connectie tussen nomadic device en BackOffice. Hiervoor heeft de methode twee parameters nodig. De BackOffice IP als een byte array en de poort waar de connectie gemaakt moet worden.
- send() zorgt ervoor dat de data werkelijk verzonden wordt. Als parameter wordt de data als String verwacht. wanneer de methode is afgelopen wordt er een boolean verstuurd met de waarde true/false. Deze waarde hangt van het feit af of de data intact is aangekomen.
- disconnect() zorgt ervoor dat de connectie met de BackOffice verbroken wordt.

De connectie die opgezet wordt met de BackOffice is een TCP connectie. Dit houdt in dat de transportlaag van het OSI model ervoor zorgt dat de data intact aankomt. Ik heb bewust gekozen dat de BackOffice alsnog een ontvangst bericht terug stuurt naar de nomadic device. Ik heb hiervoor gekozen omdat deze ontvangstbericht door gestuurd moet worden naar de ATOP.

**ConnectionUSB**

De conenctionUSB klasse wordt in de constructiefase beschreven. Het is nog niet bekend hoe deze klasse in zijn werking gaat.

## Sequentiediagram




Dit sequentiediagram spreek erg voor zich. Er wordt eerst een connectie opgezet waar dan het resultaat van wordt terug gestuurd als boolean. Wanneer dit goed is gegaan zal de data verstuurd worden. Na het versturen van de data wordt er een antwoord terug gestuurd en tot slot wanneer dit allemaal goed is gegaan zal het Android toestel de connectie sluiten.


# Spits in-vehicle koppelingsprotocol voor een mobiel apparaat

| | |
|---|---|
| Document | Ontwerprapport Iteratie 1 |
| Auteur | :Jessie Hernandez |
| Datum | 14 september 2010 |
| Versie | :4 |
| Afstudeerbedrijf | : Logica Nederland BV |
| Divisie | :Technical Software Engineering |
| Competence | : |
| Bedrijfsbegeleiders | :Bert de Neef, Edwin Essenius |
| Bedrijfsmentor | :Richard Spruit |
| Onderwijsinstelling | :De Haagse Hogeschool |
| Opleiding | :Technische informatica |
| Afstudeercoördinator | :Hans Witkamp |
| Examinator | :Cobie van der Hoek |

DE HAAGSE HOGESCHOOL

WORKING TOMORROW
TO TRY IS TO LEARN

Logica is a business and technology service company, employing 39,000 people. It provides business consulting, systems integration and outsourcing to clients around the world, including many of Europe's largest businesses. Logica creates value for clients by successfully integrating people, business and technology. It is committed to long term collaboration, applying insight to create innovative answers to clients' business needs.

Logica is listed on both the London Stock Exchange and Euronext (Amsterdam) (LSE: LOG; Euronext: LOG).

More information is available at www.logica.com.

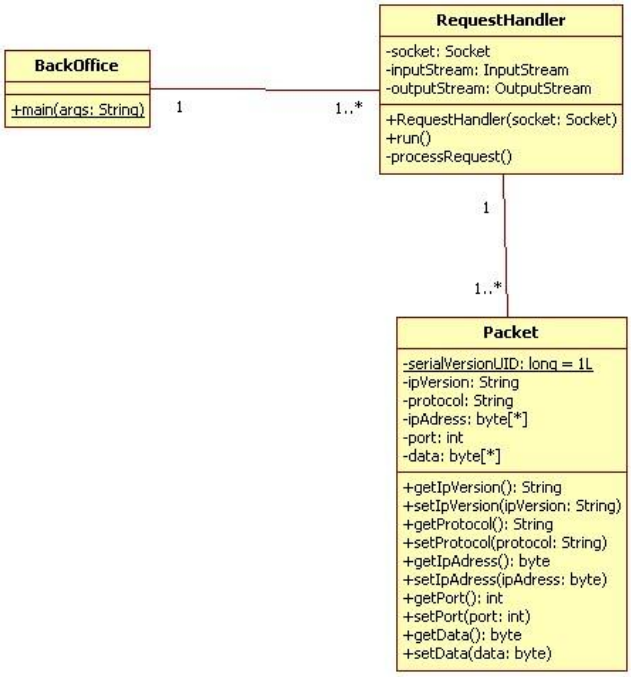
# Inleiding

In dit document wordt het ontwerp in detail besproken. Dit document is gebaseerd op de constructie fase iteratie 2. In deze fase wordt het gehele communicatielink gemoduleerd

# BackOffice

## klassendiagram

In dit hoofdstuk wordt het klassendiagram en het sequentiediagram van de BackOffice uitgelegd. Doordat er in de tweede iteratie besloten is dat er gebruik zal worden gemaakt van serializable objecten om data te versturen.




**Figuur 45: klassendiagram BackOffice Iteratie 1**

### BackOffice

Dit is de Main klasse. In deze klasse wordt de BackOffice geinitialiseerd.

### RequestHandler

De RequestHandler is de klasse die alle verzoeken die binnen komen verwerkt. In deze klasse worden de objecten ontvangen door middel van de InputStream wordt elk binnen komend Packet object gedeserializeerd. De OutputStream zorgt ervoor dat het Packet object geserializeerd wordt en verstuurd wordt.

### Attributen

- Om een connectie op te zetten wordt een Socket gebruikt, een socket is een eindpunt om twee computers te laten communiceren.
- Het attribuut input van het type InputStream ontvangt alle data die via de socket binnenkomt.
- output van het type OutputStream zorgt ervoor dat de data verzonden kan worden via het netwerk.

**Methode**

- RequestHandler is de constructor van deze klasse en verwacht als paramter de socket.
- run: deze methode is een overschreven methode uit de Runnable interface. Deze methode wordt automatisch aangeroepen bij het aanmaken van een thread deze methode zalk de prive methode processRequest aanroepen
- processRequest is de methode die ervoor zorgt dat de data ontvangen en verwerkt wordt.

**Packet**

Deze klasse is het object dat uiteindelijk verstuurd wordt. Deze klasse heeft een aantal methodes met voor elke methode een getter en een setter methode.

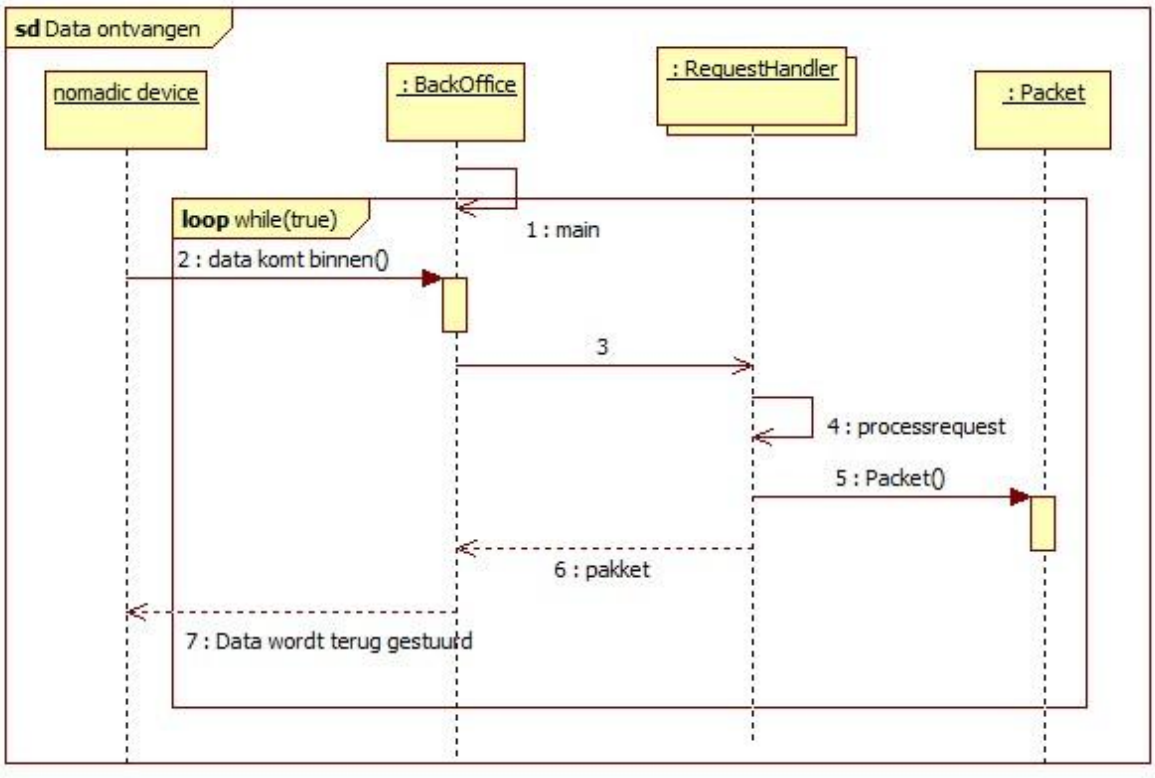
**Attributen**

- SerialVersionUID is het unieke ID van het object. Dit attribuut wordt gebruikt bij het deserialiseren om te bepalen of het object de goede versie is. Dit attribuut is een van het type long en is een final static attribuut.
- ipVersion is het attribuut wat weergeeft wat voor pakket er aankomt. Of voor het IPv6 netwerk of IPv4netwerk. De implementatie voor het IPv6 netwerk wordt niet in dit project behandeld.
- Het protocol attribuut geeft door wat voor soort connectie er gemaakt moet worden. In dit proof of concept zal er alleen met TCP pakketen gewerkt worden.
- ipAddress van het type byte array geeft het IP weer als dotted decimal notation.
- port van het type int geeft de poort weer van de BackOffice waar de connectie gemaakt moet worden.
- De byte array data bevat de data dat verstuurd moet worden. Dit kan zowel tekst als een bytestream zijn.

**Methode**

Voor elke methode is er een getter en setter.

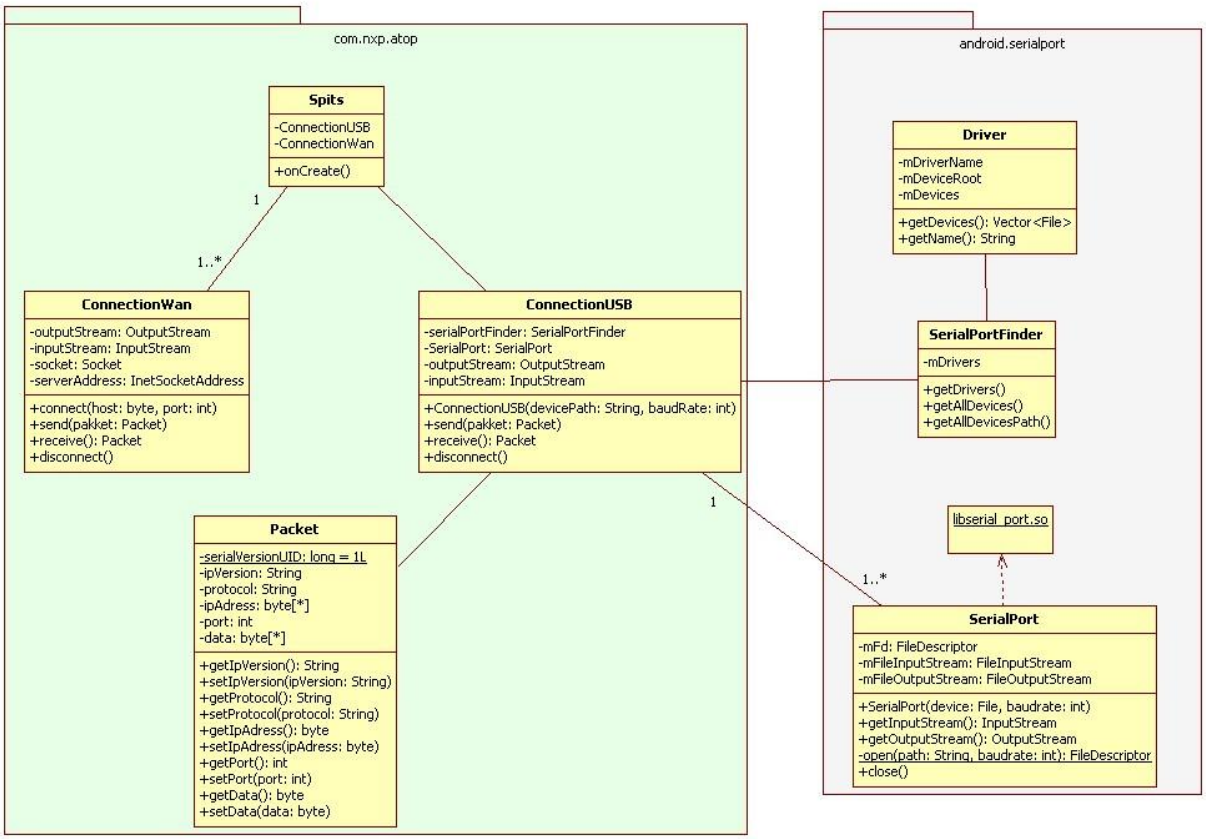
## Sequentiediagram




In deze klasse is er maar een methode. Deze methode zal de BackOffice opzetten aan de hand van een ServerSocket. Wanneer deze ServerSocket is aangemaakt zal de applicatie in een oneindige lus wachten tot er data binnen komt die voor de BackOffice bestemd is. Wanneer de data binnenkomt zal de main methode de RequestHandler aanroepen en deze in een thread stoppen om op die manier meerdere connecties aan te gaan.

# Android toestel

## klassendiagram




De klassen in de package Android.serialport zijn klassen die zijn geschreven door een andere ontwikkelaar. Om deze reden zal ik die klassen niet beschrijven. Deze klassen kunnen direct gekopieerd worden zonder enige aanpassingen aan de code van de klassen. Verder maken deze klassen gebruik van de gedeelde bibliotheek libserial_port.so.

**Spits**

Bij initialisatie van de applicatie op het Android platform zal eerst de Spits klasse worden aangeroepen. Deze klasse is een overerving van de klasse Activity. Deze klasse zal eerst een connectie met de ATOP via USB maken. Pas wanneer data binnen komt en klaar is voor het verzenden wordt de connectie met de BackOffice gemaakt.

**ConnectionWan**

De ConnectionWan klasse zorgt voor de communicatie tussen nomadic device en BackOffice. Deze klasse heeft vier attributen en drie methodes. Deze worden hieronder uitgelegd.

**Attributen**

- Het attribuut out van het type OutputStream wordt gebruikt om de data die vanuit de BackOffice komt te ontvangen.
- bufferedReader van het type BufferedReader is het attribuut die alle data ontvangt en buffert.
- socket van het type Socket wordt gebruikt om de connectie op te zetten
- address van het type InetSocket wordt gebruikt om het IP van de BackOffice op te geven.

**Methodes**

- De methode connect () zorgt voor het initialiseren van de connectie tussen nomadic device en BackOffice. Hiervoor heeft de methode twee parameters nodig. De BackOffice IP als een byte array en de poort waar de connectie gemaakt moet worden.
- send() zorgt ervoor dat de data werkelijk verzonden wordt. Als parameter wordt de data als String.
- disconnect() zorgt ervoor dat de connectie met de BackOffice verbroken wordt.

De connectie die opgezet wordt met de BackOffice is een TCP connectie. Dit houdt in dat de transportlaag van het OSI model ervoor zorgt dat de data intact aankomt. Ik heb bewust gekozen dat de BackOffice alsnog een ontvangst bericht terug stuurt naar de nomadic device. Ik heb hiervoor gekozen omdat deze ontvangstbericht door gestuurd moet worden naar de ATOP, de ATOP blijft namelijk een aantal seconden wachten op antwoord van de BackOffice.

**ConnectionUSB**

De klasse ConnectionUSB zorgt ervoor dat het mogelijk is om data via USB naar de ATOP te kunnen versturen. Deze klasse heeft 4 attributen en 4 methodes. Hieronder krijgt u uitleg van elk attribuut en elke methode

**Attribuut**

- serialPortFinder van het type SerialPortFinder is het object dat naar de seriële porten zoekt op het Android toestel.
- serialPort is het object dat eenmaal het seriele poort gevonden is waarmee gecommuniceerd gaat worden zorgt dat er gecommuniceerd kan worden.
- De outputStream zorgt ervoor dat de data geschreven wordt naar de seriele poort waar de ATOP aan gekoppeld is.
- De inputStream leest alle binnenkomende data dat via het seriele poort dat met de ATOP gekoppeld is.

**methode**

- De constructor ConnectionUSB zorgt ervoor dat de connectie gemaakt kan worden met de seriële poort die de data doorstuurt naar de ATOP verwacht als parameters het pad naar de seriële poort en de baudrate van het apparaat.
- Send is de methode die ervoor zorgt dat de data verzonden kan worden. Als paramter verwacht deze methode het object dat verzonden moet worden.
- Receive is de methode die de data ontvangt. Deze methode stuurt het object wat ontvangen is terug.
- Disconnect is de methode die ervoor zorgt dat het apparaat ontkoppeld wordt.

**Packet**

Deze klasse is het object dat uiteindelijk verstuurd wordt. Deze klasse heeft een aantal methodes met voor elke methode een getter en een setter methode.

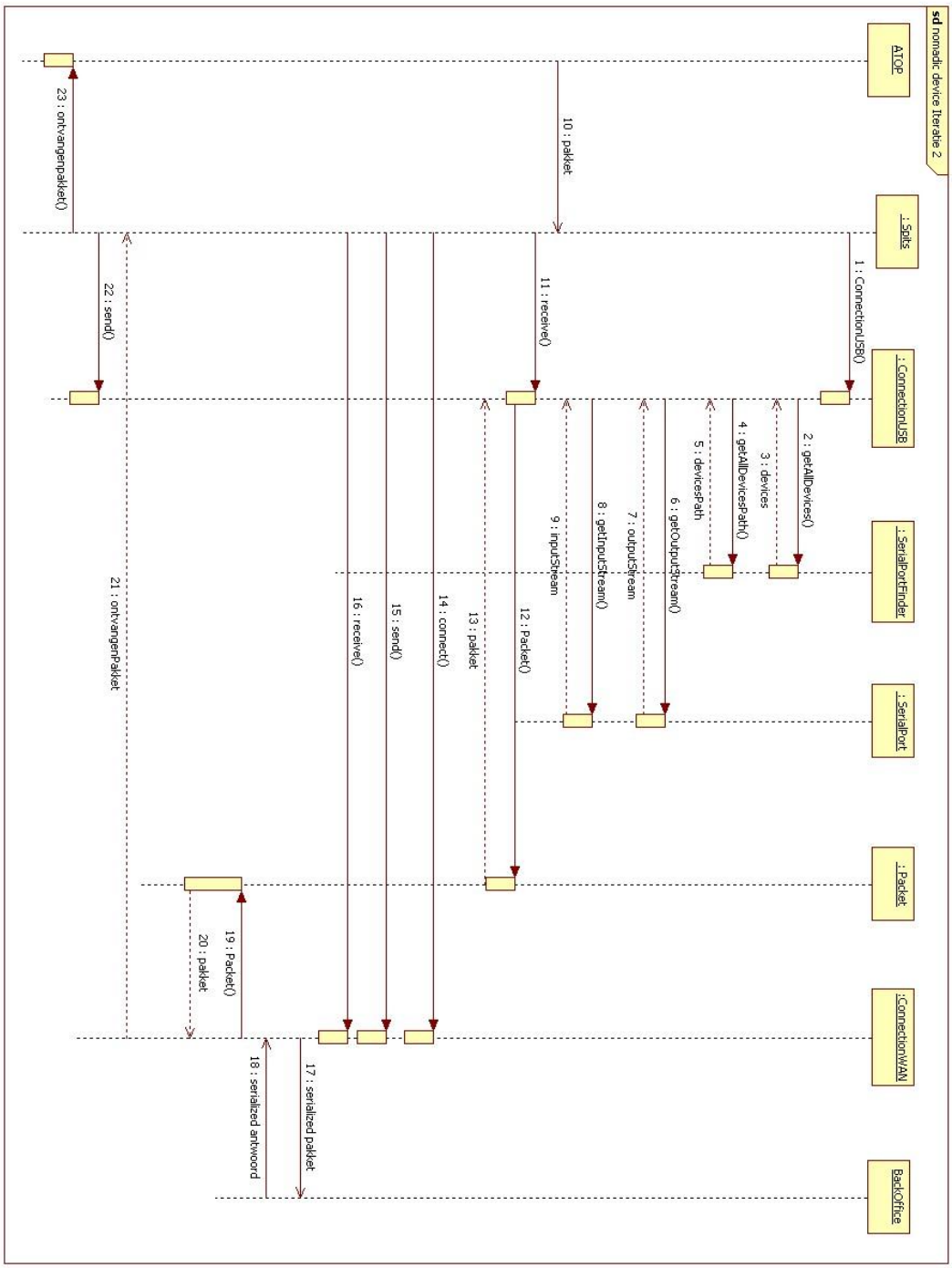
**Attributen**

- SerialVersionUID is het unieke ID van het object. Dit attribuut wordt gebruikt bij het deserialiseren om te bepalen of het object de goede versie is. Dit attribuut is een van het type long en is een final static attribuut.
- ipVersion is het attribuut wat weergeeft wat voor pakket er aankomt. Of voor het IPv6 netwerk of IPv4netwerk. De implementatie voor het IPv6 netwerk wordt niet in dit project behandeld.
- Het protocol attribuut geeft door wat voor soort connectie er gemaakt moet worden. In dit proof of concept zal er alleen met TCP pakketen gewerkt worden.
- ipAddress van het type byte array geeft het IP weer als dotted decimal notation.
- port van het type int geeft de poort weer van de BackOffice waar de connectie gemaakt moet worden.
- De byte array data bevat de data dat verstuurd moet worden. Dit kan zowel tekst als een bytestream zijn.

**Methode**

Voor elke methode is er een getter en setter.

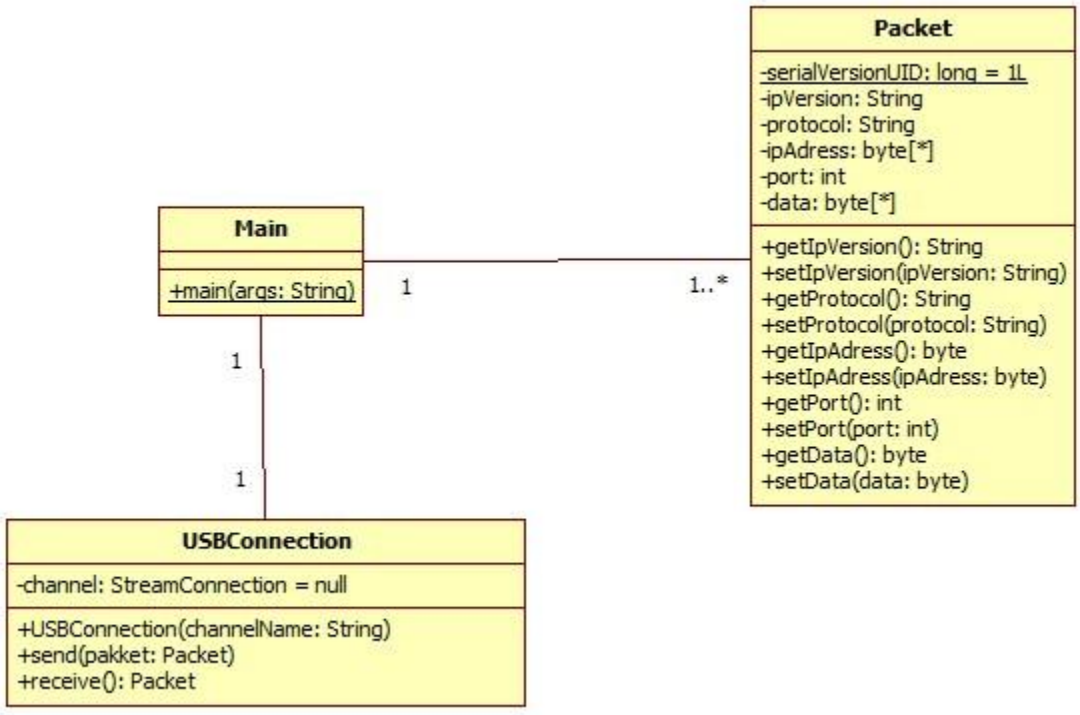
## Sequentiediagram



In dit sequentiediagram wordt de sequentie van het Android toestel weergegeven. Bij initialisatie van de applicatie wordt er eerst een USB connectie gemaakt met de ATOP. Om deze connectie te maken moet er eerst gezocht worden naar het adres dat de ATOP op het Android toestel heeft. Dit wordt gedaan met de getAllDevices methode. Wanneer alle apparaten zijn gevonden worden deze teruggestuurd naar de klasse ConnectionUSB waar dan direct wordt gevraagd voor het pad naar elk apparaat. Wanneer er een apparaat en pad gekozen is wordt de outputstream en de inputstream van het apparaat opengezet zodat er data kan worden verstuurd en ontvangen. Nadat de twee streams zijn aangemaakt wordt er gewacht op het inkomende object vanaf de ATOP. Wanneer deze binnen komt wordt het object ingelezen en worden de BackOffice IP en poort gelezen. Met deze informatie wordt er een connectie opgezet met de BackOffice nadat de connectie is opgezet wordt het object geserialiseerd en verstuurd. Na het versturen van het object gaat de applicatie wachten op antwoord van de BackOffice. Wanneer het antwoord binnenkomt worde het object door verstuurd naar de ATOP.

# ATOP

## Klassendiagram




### Main

De main klasse is de hoofdklasse van deze applicatie. In de main methode wordt het object gemaakt wat later verstuurd wordt als geserialiseerd object.

**USBConnection**

De USBConnection klasse regelt alle USB gerelateerde zaken voor de ATOP. De klasse heeft een attribuut channel van het type StreamConnetion en dit attribuut wordt gebruikt om het mux kanaal op te geven waarnaar de connectie op de ATOP gemaakt moet worden.

**Methode**

- USBConnection is de constructor van USBConnection en deze constructor zorgt voor de USB connectie met het Android toestel. De methode verwacht als parameter de kanaalnaam waar naar toe er geconnecteerd wordt. De kanaalnaam is van het type String.
- De send methode is de methode die de data verstuurt. De methode serialiseert het object eerst en stuurt deze vervolgens in stukjes van 60 bytes naar het Android toestel
- Deze methode ontvangt alle binnenkomende objecten. Wanneer deze objecten binnen komen worden ze eerst gedeserialiseerd. En dan verwerkt.

**Packet**

Deze klasse is het object dat uiteindelijk verstuurd wordt. Deze klasse heeft een aantal methodes met voor elke methode een getter en een setter methode.

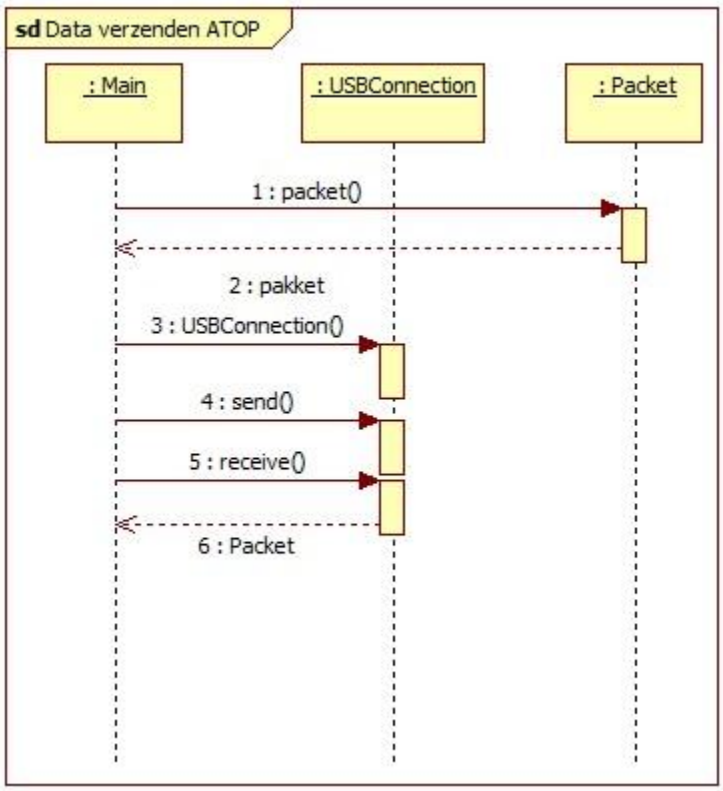
**Attributen**

- SerialVersionUID is het unieke ID van het object. Dit attribuut wordt gebruikt bij het deserialiseren om te bepalen of het object de goede versie is. Dit attribuut is een van het type long en is een final static attribuut.
- ipVersion is het attribuut wat weergeeft wat voor pakket er aankomt. Of voor het IPv6 netwerk of IPv4netwerk. De implementatie voor het IPv6 netwerk wordt niet in dit project behandeld.
- Het protocol attribuut geeft door wat voor soort connectie er gemaakt moet worden. In dit proof of concept zal er alleen met TCP pakketen gewerkt worden.
- ipAddress van het type byte array geeft het IP weer als dotted decimal notation.
- port van het type int geeft de poort weer van de BackOffice waar de connectie gemaakt moet worden.
- De byte array data bevat de data dat verstuurd moet worden. Dit kan zowel tekst als een bytestream zijn.

**Methode**

Voor elke methode is er een getter en setter.

## Sequentiediagram




In dit sequentiediagram is de interactie van de ATOP gemoduleerd. Als eerste wordt het pakket aangemaakt door een object te maken. Nadat het object gemaakt is wordt er een USB connectie vanaf de klasse USBConnection gemaakt met het Android toestel. Nadat de connectie gemaakt is wordt het packet geserialiseerd en verstuurd naar het Android toestel. Na het versturen van de data gaat de ATOP wachten op antwoord van de BackOffice.

v


# Spits in-vehicle koppelingsprotocol voor een mobiel apparaat

| | |
|---|---|
| Document | Testplan |
| Auteur | :Jessie Hernandez |
| Datum | :14 september |
| Versie | 0.2 |
| Afstudeerbedrijf | : Logica Nederland BV |
| Divisie | : |
| Competence | : |
| Bedrijfsbegeleiders | :Bert de Neef, Edwin Essenius |
| Bedrijfsmentor | :Richard Spruit |
| Onderwijsinstelling | :De Haagse Hogeschool |
| Opleiding | :Technische informatica |
| Afstudeercoördinator | :Hans Witkamp |
| Examinator | :Cobie van der Hoek |

DE HAAGSE HOGESCHOOL

WORKING TOMORROW
TO TRY IS TO LEARN

Logica is a business and technology service company, employing 39,000 people. It provides business consulting, systems integration and outsourcing to clients around the world, including many of Europe's largest businesses. Logica creates value for clients by successfully integrating people, business and technology. It is committed to long term collaboration, applying insight to create innovative answers to clients' business needs.

Logica is listed on both the London Stock Exchange and Euronext (Amsterdam) (LSE: LOG; Euronext: LOG).

More information is available at www.logica.com.

## Versie Beheer

| Versie | Datum | Auteur | veranderingen |
|---|---|---|---|
| 0.1 | 13-9-2010 | jessie | concept |
| 0.2 | 14-9-2010 | Jessie | Concept aangepast en Testframe toegepast |

# Inhoudsopgave

# Inleiding

Dit document is een testplan voor het testen van de communicatielink tussen de ATOP en de BackOffice gebruikmakende van een nomadic device. in dit document wordt beschreven hoe elk onderdeel van het systeem wordt getest en welke tests er daarvoor zijn uitgekozen. Verder wordt de planning voor het testen beschreven en hoe er verwacht wordt te gaan testen.

# Doel

In dit hoofdstuk zullen we het hebben over de doel van dit testplan. Er zal eerst wat informatie worden gegeven over het project en alle belanghebbenden.

## Doel van het project

De doel van dit project is het maken van een communicatielink tussen een OBU en de BackOffice door gebruik te maken van de communicatie mogelijkheden van een mobiele apparaat. Hiervoor wordt gebruik gemaakt van USB als communicatiekanaal tussen de OBU en het mobiele apparaat. Wanneer de internetlink uitvalt (de link tussen het mobiele apparaat en de BackOffice) zal een melding naar de applicatie worden gestuurd dat de connectie is uitgevallen en automatische opnieuw worden geinitialiseerd.

## Doel testplan

De doel van het testplan is het informeren van de test- en ontwikkel teams die bij het test proces van de communicatielink betrokken zijn over de aanpak, activiteiten en te op te leveren producten aan het eind van de testfase.

Het testplan is geschreven aan de hand van de testmethode Testframe.

Bij het schrijven van het testplan zijn er enkele mensen bij betrokken. De auteur van het testplan is jessie hernandez, en de goedkeurende is de opdrachtgever Richard Spruit.

# Opdrachtomschrijving

De opdrachtgever voor dit project is mijn Spits begeleider. De opdrachtnemer van dit project om te testen is de student. Het systeem dat getest wordt is de middlewarelaag geschreven voor een Android apparaat. Deze laag is geschreven in JAVA. Zie hoofdstuk doel voor verdere informatie over het project.

## Scope

Tot de scope van deze testplan behoren de volgende punten

- De nomadic device code
- De ATOP code (JAVA gedeelte)

Niet tot de scope van deze testplan behoort de geleverde hardware. Dit houdt in het programmatuur die op de ATOP draait die door de fabrikant wordt geleverd, de hardware van de ATOP, het inladen van de 2 Android kernelmodules voor het aanzetten van USB host en USB Serieel en de BackOffice code.

## precondities and aannames

- Het testen moet op 13 september 2010 beginnen en moet klaar zijn op 17 september 2010.
- De ontwikkelteams zijn klaar met de constructie van het systeem.
- Het testplan moet door alle belanghebbenden worden geaccepteerd
- De test omgeving is op tijd klaar.
- Het testen van de testcases begint niet voordat de precondities zijn gehaald

## acceptatiecriteria

in het onderstaande tabel ziet u de acceptatie criteria voor de communicatielink en waaraan ze worden getest.

| **criterium** | **standaard** |
|---|---|
| De data wordt intact verstuurd en ontvangen door alle onderdelen van het systeem. | Er is voldaan aan de bijhorende eisen. |
| er is foutafhandeling geïmplementeerd | Er is voldaan aan de bijhorende eisen. |

## Testproducten

Aan het einde van de testfase zullen er een aantal producten moeten worden opgeleverd de producten die worden opgeleverd zijn de testgevallen het testrapport met uitkomst van alle testgevallen en de acceptatietest met oordeel van de opdrachtgever.

# Documentatie

## Basis voor het testplan

Voor het schrijven van het testplan zijn de volgende documenten gebruikt. De plan van aanpak/afstudeerplan geschreven door Jessie Hernandez. En de architectuurrapport geschreven door Jessie Hernandez

## methode

voor het schrijven van de testplan is het boek Testframe een praktische handleiding bij het testen van informatiesystemen.

# Testsoorten

In dit hoofdstuk wordt beschreven wat er getest zal worden, hoe diep elke test wordt uitgevoerd en wanneer elke test wordt uitgevoerd. Om de eisen van de communicatielink te testen zal er een systeemtest worden uitgevoerd en daarna zal er ook een acceptatietest worden uitgevoerd. Deze test wordt door de opdrachtgever uitgevoerd.

## Testtechnieken

Om de tests volledig te kunnen uitvoeren ga ik gebruik maken van de testtechnieken real-life test en error guessing test. Deze tests zijn gekozen om de eisen en wensen van de opdrachtgever naast de testgevallen te leggen en uit te voeren.

# Aanpak

## Error-guessing

Om deze testtechniek uit te voeren moet er gekeken worden naar wat de zwakke plekken van het systeem zijn. Zwakke plekken kunnen in een systeem kunnen bijvoorbeeld foutafhandeling zijn.

Wanneer er illegale waarden worden opgestuurd het testen op illegale waarden is echter niet van toepassing op het te testobject doordat alle data die de applicatie gebruikt zelf wordt gegenereerd.

Onderdelen van het systeem die meerdere malen onderdeel waren van wijzigingsverzoeken. Beveiliging en het claimen van teveel resources.

## Real-Life test

In de real-life test wordt getest in een zoveel mogelijk realistische omgeving. Om deze test uit te voeren moet de test dan worden uitgevoerd zoals die in productie omgeving zou moeten gaan werken. De concrete testomgeving word in de testgevallen beschreven.

# Organisatie

## Rollen, taak en verantwoordelijkheden

| Rol | Persoon | Taak |
|---|---|---|
| Tester | Jessie Hernandez | - produceren van testgevallen<br>- tests uitvoeren<br>- schrijven van de testresultaten<br>- schrijven van testplan |

# infrastructuur

## Test omgeving

De tests zullen in de ontwikkel omgeving worden uitgevoerd.

# Management

## omgevingmanagement

De testomgeving voor de tests wordt gemanaged door Logica. Deze omgeving zal hetzelfde zijn als e ontwikkel omgeving.

## fouten reportage

Om alle fouten en bugs bij te houden wordt er een buglijst bijgehouden. Deze buglijst is te vinden in het testrapport met de uitkomst van alle testcases.

# Globale Planning

## Planning

Hieronder ziet u de planning die is opgesteld voor de testfase.

*Day 1-3*
Error Guessing
*Day 3-5*
Real-Life

## Mijlpalen

De mijlpalen voor de testfase kunt u in de tabel hieronder vinden.

| Mile stone description | Date |
|---|---|
| **Error guessing test** | **15 september 2010** |
| **Real life test** | **17 september 2010** |


# Spits in-vehicle koppelingsprotocol voor een mobiel apparaat

| | |
|---|---|
| Document | Test gevallen |
| Auteur | :Jessie Hernandez |
| Datum | 14 september 2010 |
| Versie | :4 |
| Afstudeerbedrijf | : Logica Nederland BV |
| Divisie | :Technical Software Engineering |
| Competence | : |
| Bedrijfsbegeleiders | :Bert de Neef, Edwin Essenius |
| Bedrijfsmentor | :Richard Spruit |
| Onderwijsinstelling | :De Haagse Hogeschool |
| Opleiding | :Technische informatica |
| Afstudeercoördinator | :Hans Witkamp |
| Examinator | :Cobie van der Hoek |

DE HAAGSE HOGESCHOOL

WORKING TOMORROW
TO TRY IS TO LEARN

Logica is a business and technology service company, employing 39,000 people. It provides business consulting, systems integration and outsourcing to clients around the world, including many of Europe's largest businesses. Logica creates value for clients by successfully integrating people, business and technology. It is committed to long term collaboration, applying insight to create innovative answers to clients' business needs.

Logica is listed on both the London Stock Exchange and Euronext (Amsterdam) (LSE: LOG; Euronext: LOG).

More information is available at www.logica.com.

# Testgevallen Nomadic device – BackOffice (iteratie 1)

## Error Guessing

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan. |
| Taken: | 5. Start de applicatie op de mobiele apparaat.<br>6. Druk op de knop verzenden. |
| Verwachtte uitvoer: | De tekst “Welcome to the spits open platform” verschijnt op het scherm bij de BackOffice er verschijnt een label onder de button met de juist verstuurde tekst als hoofdletters. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

## Real-life test

**Profielschets**

Om de real-life test zo realistisch mogelijk te maken is er profielschets vereist. De profielschets zal de situatie beschrijven waarin getest zal worden. In de volgende paragraaf vind u de situatie waarin de real-life test zal worden uitgevoerd.

De test zal mobiel moeten plaatsvinden doordat de OBU in een auto zal worden geplaatst. Er wordt echter gekozen om de test toch niet in een auto uit te voeren omdat dit geen toegevoegde waarde heeft op de test. De test zal representatief zijn omdat de data vanaf de OBU tot het mobiele apparaat via USB verloopt en dus geen interferentie ondervindt.

Bij de connectie tussen het mobiele apparaat en de BackOffice zal de sessie niet onderbroken worden bij het veranderen van zendmast. De IP-adressen zijn namelijk statische adressen. De implementatie van het mobiele internet is echter per provider verschillend en is dus provider afhankelijk of het IP dynamisch of statisch is.

De tests worden uitgevoerd bij T-Mobile als provider. Na contact op te nemen met de technische dienst van T-Mobile is gebleken dat T-Mobile gebruik maakt van statische IP-adressen. Dit betekent dat sessies gewoon open blijven.

Als omgeving wordt daarom gekozen om de tests in de ontwikkel omgeving te doen. Een andere reden voor deze keuze is de mogelijkheid om de debug board van de OBU te koppelen om zo ook de systeem berichten van de OBU te kunnen lezen.

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan. |
| Taken: | 3. Start de applicatie op de mobiele apparaat.<br>4. Druk 5x op de knop verzenden. |
| Verwachtte uitvoer: | De tekst “Welcome to the spits open platform” verschijnt5 maal op het scherm bij de BackOffice er verschijnt een label onder de button met de juist verstuurde tekst als hoofdletters. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

# Testgevallen communicatielink (Iteratie 2)

## Error Guessing

Als mogelijke zwakke plekken in de communicatielink heb ik bedacht dat het verzenden en ontvangen van data via het USB kanaal problemen kan opleveren. Een tweede zwakke plek is de foutafhandeling wanneer er wat mis gaat in de applicatie.

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 4. Start de applicatie op de mobiele apparaat.<br>5. Sluit de stroom stekker aan.<br>6. Als er gevraagd wordt om root rechten kies “allow”* |
| Verwachtte uitvoer: | De data op de ATOP binnengekomen met als verandering dat de gestuurde data als hoofdletters terug is gestuurd. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

| Naam: | Connectie verbreken |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 9. Start de applicatie op de mobiele apparaat.<br>10. Sluit de stroom stekker aan.<br>11. Als er gevraagd wordt om root rechten kies “allow”*<br>12. Na 5 secondes trek de stekker uit het mobiele apparaat. |
| Verwachtte uitvoer: | Er verschijnt een melding op de mobiele apparaat met de boodschap dat de data niet kon worden verzonden. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

## Testgevallen real-life test

| | |
|---|---|
| Naam: | De applicatie stuurt de data synchroon, er moet altijd een response vanuit de ATOP naar de applicatie. |
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 9. Start de applicatie op de mobiele apparaat.<br>10. Sluit de stroom stekker aan.<br>11. Als er gevraagd wordt om root rechten kies “allow”*<br>12. Kijk op de pc in het programma spyTracer naar het outputscherm. |
| Verwachtte uitvoer: | De applicatie op de OBU blijft ‘hangen’ tot wanneer er data terug gestuurd is vanaf de BackOffice/mobiele apparaat. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

| | |
|---|---|
| Naam: | De link moet automatisch opgezet worden zonder input van de gebruiker. |
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 7. Start de applicatie op de mobiele apparaat.<br>8. Sluit de stroom stekker aan.<br>9. Als er gevraagd wordt om root rechten kies “allow”* |
| Verwachtte uitvoer: | De connectie wordt opgezet en de data wordt verstuurd van OBU naar mobiele apparaat naar BackOffice en terug. |
| Werkelijke uitvoer | |
| Opmerkingen: | |

*er kan ook gekozen worden voor “always allow”.


# Spits in-vehicle koppelingsprotocol voor een mobiel apparaat

| | |
|---|---|
| Document | Test Rapport |
| Auteur | :Jessie Hernandez |
| Datum | 14 september 2010 |
| Versie | :4 |
| Afstudeerbedrijf | : Logica Nederland BV |
| Divisie | :Technical Software Engineering |
| Competence | : |
| Bedrijfsbegeleiders | :Bert de Neef, Edwin Essenius |
| Bedrijfsmentor | :Richard Spruit |
| Onderwijsinstelling | :De Haagse Hogeschool |
| Opleiding | :Technische informatica |
| Afstudeercoördinator | :Hans Witkamp |
| Examinator | :Cobie van der Hoek |

DE HAAGSE HOGESCHOOL

WORKING TOMORROW
TO TRY IS TO LEARN

Logica is a business and technology service company, employing 39,000 people. It provides business consulting, systems integration and outsourcing to clients around the world, including many of Europe's largest businesses. Logica creates value for clients by successfully integrating people, business and technology. It is committed to long term collaboration, applying insight to create innovative answers to clients' business needs.

Logica is listed on both the London Stock Exchange and Euronext (Amsterdam) (LSE: LOG; Euronext: LOG).

More information is available at www.logica.com.

# Inleiding

In dit rapport is het testrapport beschreven. U kunt alle testgevallen doorlopen met hun bijbehorende uitkomsten. Aan het eind van het document krijgt u een lijst met bugs die tijdens het testen geconstateerd zijn.

# Testgevallen Nomadic device – BackOffice (iteratie 1)

## Error Guessing

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan. |
| Taken: | 7. Start de applicatie op de mobiele apparaat.<br>8. Druk op de knop verzenden. |
| Verwachtte uitvoer: | De tekst “Welcome to the spits open platform” verschijnt op het scherm bij de BackOffice er verschijnt een label onder de button met de juist verstuurde tekst als hoofdletters. |
| Werkelijke uitvoer | De label presenteert de tekst zoals het hoort en de BackOffice ontvangt de data ook intact. |
| Opmerkingen:<br><br>geen | |

## Real-life test

**Profielschets**

Om de real-life test zo realistisch mogelijk te maken is er profielschets vereist. De profielschets zal de situatie beschrijven waarin getest zal worden. In de volgende paragraaf vind u de situatie waarin de real-life test zal worden uitgevoerd.

De test zal mobiel moeten plaatsvinden doordat de OBU in een auto zal worden geplaatst. Er wordt echter gekozen om de test toch niet in een auto uit te voeren omdat dit geen toegevoegde waarde heeft op de test. De test zal representatief zijn omdat de data vanaf de OBU tot het mobiele apparaat via USB verloopt en dus geen interferentie ondervindt.

Bij de connectie tussen het mobiele apparaat en de BackOffice zal de sessie niet onderbroken worden bij het veranderen van zendmast. De IP-adressen zijn namelijk statische adressen. De implementatie van het mobiele internet is echter per provider verschillend en is dus provider afhankelijk of het IP dynamisch of statisch is.

De tests worden uitgevoerd bij T-Mobile als provider. Na contact op te nemen met de technische dienst van T-Mobile is gebleken dat T-Mobile gebruik maakt van statische IP-adressen. Dit betekent dat sessies gewoon open blijven.

Als omgeving wordt daarom gekozen om de tests in de ontwikkel omgeving te doen. Een andere reden voor deze keuze is de mogelijkheid om de debug board van de OBU te koppelen om zo ook de systeem berichten van de OBU te kunnen lezen.

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan. |
| Taken: | 3. Start de applicatie op de mobiele apparaat.<br>4. Druk 5x op de knop verzenden. |
| Verwachtte uitvoer: | De tekst “Welcome to the spits open platform” verschijnt5 maal op het scherm bij de BackOffice er verschijnt een label onder de button met de juist verstuurde tekst als hoofdletters. |
| Werkelijke uitvoer | De label presenteert de data zoals het hoort en de BackOffice heeft de data ook 5 maal ontvangen. |
| Opmerkingen:<br><br>geen | |

# Testgevallen communicatielink (Iteratie 2)

## Error Guessing

Als mogelijke zwakke plekken in de communicatielink heb ik bedacht dat het verzenden en ontvangen van data via het USB kanaal problemen kan opleveren. Een tweede zwakke plek is de foutafhandeling wanneer er wat mis gaat in de applicatie.

| Naam: | Data versturen |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 4. Start de applicatie op de mobiele apparaat.<br>5. Sluit de stroom stekker aan.<br>6. Als er gevraagd wordt om root rechten kies “allow”* |
| Verwachtte uitvoer: | De data op de ATOP binnengekomen met als verandering dat de gestuurde data als hoofdletters terug is gestuurd. |
| Werkelijke uitvoer | `StreamCorruptedException/ InvalidClassException` |
| Opmerkingen:<br><br>`Dit komt door het deserialiseren van het object. Doordat de data niet goed aankomt door nog onbekende redenen.` | |

| Naam: | Connectie verbreken |
|---|---|
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 13. Start de applicatie op de mobiele apparaat.<br>14. Sluit de stroom stekker aan.<br>15. Als er gevraagd wordt om root rechten kies “allow”*<br>16. Na 5 secondes trek de stekker uit het mobiele apparaat. |
| Verwachtte uitvoer: | Er verschijnt een melding op de mobiele apparaat met de boodschap dat de data niet kon worden verzonden. |
| Werkelijke uitvoer | `Klopt. Er verschijnt een boodschap dat de connectie niet gelukt is.` |
| Opmerkingen:<br><br>`De boodschap verschijnt ook als de connectie wel gelukt is maar de data niet verzonden is` | |

## Testgevallen real-life test

| | |
|---|---|
| Naam: | De applicatie stuurt de data synchroon, er moet altijd een response vanuit de ATOP naar de applicatie. |
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 13. Start de applicatie op de mobiele apparaat.<br>14. Sluit de stroom stekker aan.<br>15. Als er gevraagd wordt om root rechten kies “allow”*<br>16. Kijk op de pc in het programma spyTracer naar het outputscherm. |
| Verwachtte uitvoer: | De applicatie op de OBU blijft wachten tot wanneer er data terug gestuurd is vanaf de BackOffice/mobiele apparaat. |
| Werkelijke uitvoer | De applicatie blijft wachten maar er komt geen data terug |
| Opmerkingen:<br><br>de data komt niet goed aan op het Android toestel. Dit komt nog door onbekende redenen. | |

| | |
|---|---|
| Naam: | De link moet automatisch opgezet worden zonder input van de gebruiker. |
| Precondities: | - Mobiele apparaat staat aan.<br>- Kernel modules zijn geladen (ehci-hcd.ko en usbserial.ko)*<br>- De ATOP en het mobiele apparaat zijn gekoppeld. |
| Taken: | 10. Start de applicatie op de mobiele apparaat.<br>11. Sluit de stroom stekker aan.<br>12. Als er gevraagd wordt om root rechten kies “allow”* |
| Verwachtte uitvoer: | De connectie wordt opgezet en de data wordt verstuurd van OBU naar mobiele apparaat naar BackOffice en terug. |
| Werkelijke uitvoer | De connectie wordt automatisch opgezet. Maar de data blijft bij de OBU hangen. |
| Opmerkingen:<br><br>De data komt niet goed aan op het Android toestel door nog onbekende redenen. | |

*er kan ook gekozen worden voor “always allow”.

# Bugs

In dit hoofdstuk vind u alle bugs die geconstateerd zijn tijdens het testen. Deze bugs zijn opgesomd met de oorzaak erbij en eventuele oplossing.

| **Bug** | **Oorzaak** | **oplossing** |
|---|---|---|
| De data van de OBU naar de Nomadic Device wordt niet altijd goed verstuurd | De buffers in de firmware raken overvol | Met een nieuwere firmware testen of de fabrikant vragen om de buffers groter te maken |
| Als de OBU data verzend blijft de nomadic device soms hangen | De ontvangst van de nomadic device weet niet hoeveel bytes er verwacht zijn dus hij blijft voor altijd wachten | Nog geen oplossing. |