Software architectuur document voor Call-a-Car

15 april 2018

Project	Call-a-Car
Author 1	Marco Huijben
Student number	838316640
Author 2	Ivo Willemsen
Student number	851926289
Date	15-4-2018
Version	1.0

Tabel 1: Metadata

Version	*Date*	*Change*
0.1	4-3-2018	Initial version
0.2	21-3-2018	Architectuur visie en scenario's
1.0	15-4-2018	Verwerken review en toevoegen views

Tabel 2: Version history

Marco Huijben	*Stakeholders and their concerns, Architecturual vision, Physical Viewpoint*
	Verwerken verbeteringen a.d.h.v. feedback, Document structuur
Ivo Willemsen	*Introduction, Requirements, Architecturual vision, Logical Viewpint,*
	Process Viewpoint, Development Viewpoint, Scenario

Tabel 3: Division of work

Inhoudsopgave

Lijst van figuren

1 Introduction

In de Automatiseringsgids deed de Minister van Verkeer uit de doeken dat ze graag een experiment wil uitvoeren met zelfrijdende auto's. Geïnspireerd door dit interview is opdrachtgever Alex Aanvoerder met het idee gekomen om een systeem te ontwikkelen dat klanten in staat stelt om op een efficiëntere manier om te gaan met het concept van autobezit. Hierbij kunnen klanten met één click in een app een zelfrijdende auto bestellen en betalen zij alleen voor de gereden kilometers. Om een groter maatschappelijk draagvlak te creëren, zijn er contacten gelegd met Bits For Freedom, die graag aan het project wil meewerken om privacy van klanten te showcasen. Vanuit de financiële hoek heeft een bank interesse om als sponsor op te treden en heeft een autofabrikant hun zelfrijdende elektrische auto's aangeboden.

Het project behelst de creatie van een systeem dat als doel heeft het flexibiliseren van het concept van autobezit m.b.v. het het gebruik van zelfrijdende auto's. Het moet mensen aanzetten om een auto te zien als een gebruiksvoorwerp en niet direct als een vorm van bezit. Indien men een auto voor een bepaalde tijd nodig heeft, kan men m.b.v. het systeem een auto configureren, een afleveradres en -tijdstip opgeven. De zelfrijdende auto zal vervolgens de klant ophalen en brengen naar de plek van bestemming, waarbij de klant alleen zal betalen voor de gereden kilometers. Nadat de klant heeft betaald, zal de zelfrijdende auto terugrijden naar een publieke parkeergarage, waar de auto zal wachten op een verzoek van een volgende klant.

1.1 Context

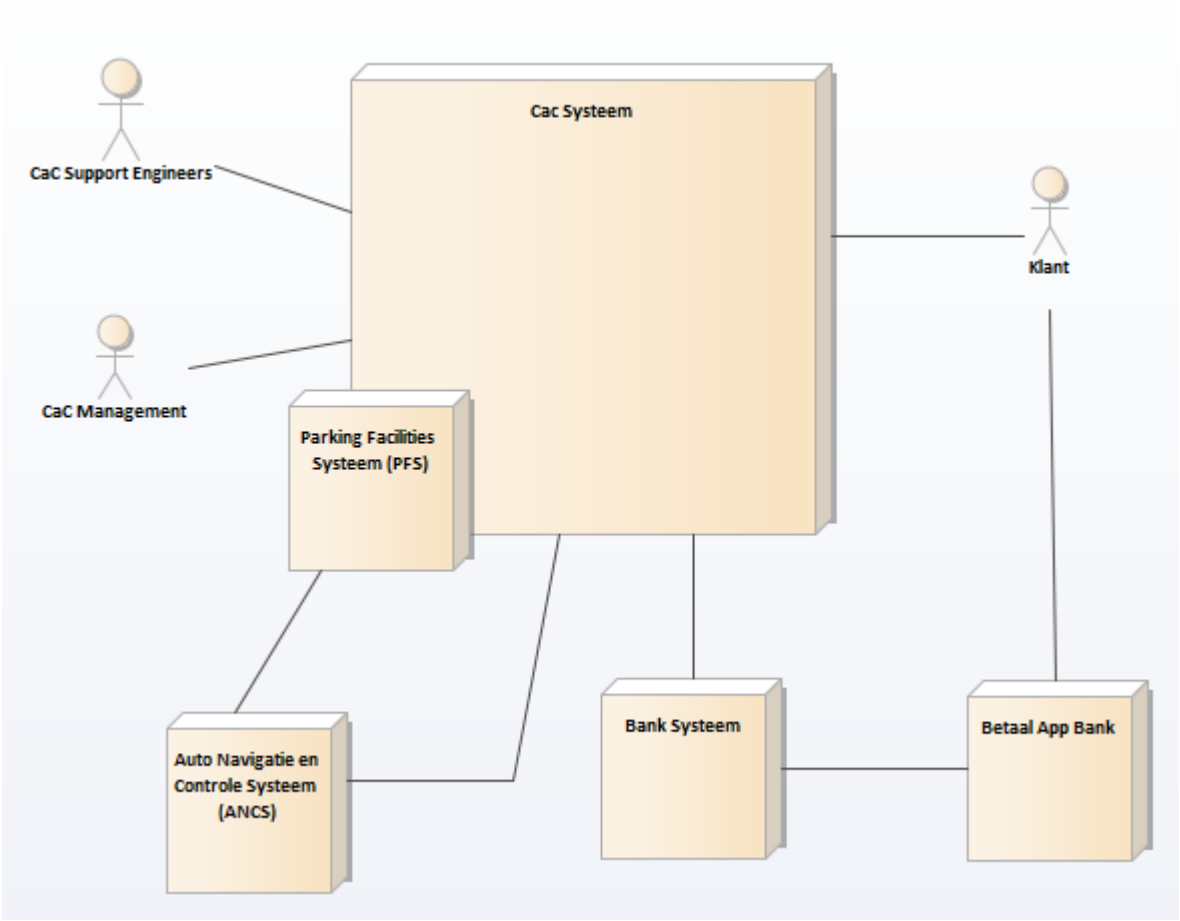
Dit project staat niet op zich zelf, maar kent vele betrokkenen. Belangrijk is om tussen al wensen, eisen en belangen van de betrokkenen een goede balans te vinden. De Minister van Verkeer wil, zoals eerder gezegd, een experiment uitvoeren met zelfrijdende auto's. Het Ministerie van Verkeer zal de minister hierin vertegenwoordigen.

De autofabrikant levert de zelfrijdende elektrische voertuigen en de software voor in de voertuigen. Privacy is tegenwoordig een hot item en de organisatie Bits Of Freedom wil dit goed geregeld zien. De bank wil zijn betalings app promoten. De provincie van Limburg levert de parkeerfaciliteiten.
Er bestaat reeds een systeem voor een flexibele autobezit, genaamd GroeneWielen, die mensen in staat stelt om op een flexibele manier op te gaan met autobezit, echter, deze oplossing is niet gebruikersvriendelijk en er zijn hoge kosten voor de klant mee gemoeid. Ook wordt bij deze oplossing geen gebruik gemaakt van zelfrijdende auto's. Het gebruikt van zelfrijdende auto's zal een kosten-reducerend effect tot gevolg hebben. GroeneWielen kan als concurrent beschouwd worden.

Een andere doelgroep zijn mensen die graag de beschikking willen hebben over een tweede auto, maar deze auto maar sporadisch gebruiken. Voor hen zou het een mooie oplossing zijn, indien ze gebruik kunnen maken van een flexibele dienst zoals Call-a-Car.

1.2 System boundaries

In figuur 1 worden de grenzen van het te ontwikkelen systeem, blok CaC Systeem, aangegeven. De volgende deelsystemen



Figuur 1: System boundaries diagram

spelen een rol in de communicatie met het Cac Systeem, maar vallen niet binnen de systeem boundaries van het te ontwikkelen systeem:

- Het "Auto Navigatie en Control Systeem". Deze software zal door de leverancier worden ontwikkeld, waarbij Call-a-Car zal aangeven wat de interface moet zijn.
- Het banksysteem. De bank bepaalt de interface en geeft aan op welke wijze het Cac systeem met het systeem van de bank kan communiceren.
- Het deelsysteem "Betaal App" van de bank. De klant zal deze app gebruiken om maandelijkse betalingen uit te voeren.
- De "Parking Facilities Systeem" (PFS) is in eerste instantie een onderdeel van het "CaC systeem" maar moet in een latere instantie losgekoppeld kunnen worden. De interfaces met PFS wordt door Call-a-Car opgesteld.

In het diagram zijn een aantal notities opgenomen die de architectuurpatronen aanduiden (AP) die in een later hoofdstuk zullen worden beschreven.

1.3 Goal of the document

Dit document heeft als doel inzicht te geven in de achtergronden van het project. Tevens wil dit document de stakeholders een indruk te geven wat hun positie in het project is en in welke manieren hun behoeften worden ingewilligd. De laatste doel is om de stakeholders duidelijk te maken welke Software Architectuur voor dit systeem toegepast gaat worden.

In hoofdstuk 2 worden de stakeholders van het systeem en hun concerns omtrent het systeem beschreven. Hieruit

worden in hoofdstuk 3 de functional requirements en de non-functional requirements opgesteld. De constraints worden ook in dit hoofdstuk uiteengezet. Aan de hand van de requirements beschrijft hoofdstuk 4 vanuit verschillende perspectieven de architectuur van het systeem. Een aantal scenario's worden opgesteld in hoofdstuk 5. De glossary, hoofdstuk 5.1.2, bevat een lijst met verklaringen van belangrijke begrippen.

2 Stakeholders and their concerns

De stakeholders voor het systeem zijn onder te verdelen in de volgende groepen:

- Opdrachtgevers (2.1)
- Ontwikkelaars (2.2)
- Gebruikers (2.3)
- Beheerders (2.4)
- Overig (2.5)

Voor elke Stakeholder wordt aangeven, indien bekend, wie zij zijn en welke relatie zij met het project hebben. Ook worden hun concerns beschreven.

2.1 Stakeholdergroep Opdrachtgevers

Deze paragraaf beschrijft de stakeholders van de groep opdrachtgevers.

Naam	Doris Dorknoper
Functie	Representant Ministerie van Verkeer
Omschrijving	Ministerie wil een experiment uitvoeren met zelfrijdende auto's
Concerns	• Het ministerie wil diverse experimenten uitvoeren met zelfrijdende auto's. • Het ministerie wil een vrije markt creëren, zodat andere bedrijven in de toekomst ook hun diensten kunnen aanbieden. Het ministerie wil niet dat het gehelen systeem van parkeerfaciliteiten en het systeem dat door Call-a-Car gebouwd wordt, in handen komt van één partij. • Het ministerie wil dat de parkeerfaciliteiten later worden verkocht aan een andere partij. • Het ministerie wil eigenaar worden van de standaard op basis welke de API zal worden gebouwd. • Het ministerie wil per week geaggregeerde informatie over de gereden kilometers. De kilometers voor het rijden naar en vanaf de parkeerplaatsen tellen hierbij niet mee. Met behulp van deze gegevens kunnen zij bepalen of het project een succes is. • Het ministerie wil per direct een melding over incidenten met de auto's. Dit geldt alleen als de auto zelfstandig rijdt. • Vanwege het budget en de tijd is het van belang dat het personeelsverloop laag is, zodat minder tijd aan inwerken besteed hoeft te worden.

Tabel 4: Stakeholder Doris Dorknoper

Naam	Alex Aanvoerder
Functie	Oprichter Call-a-Car
Omschrijving	Hij is de initiatiefnemer van het project en eigenaar van het bedrijf Call-a-Car
Concerns	• Een andere kijk op eigenaarschap van auto's. Door dit project hoeft de gebruiker geen auto te bezitten, maar bestelt naar behoefte een auto. • Alex wil dat de gebruikers tevreden zijn, zodat het experiment een gevolg krijgt. • Budget van EURO 2.000.000, waarvan EURO 1.000.000 voor o.a. reclame en implementatie. Een deel van het budget is een buffer voor het derde jaar. Voor software ontwikkeling moet onder de EURO 1.000.000 blijven. • Het project moet een werkbaar systeem als resultaat hebben en binnen twee jaar operationeel zijn. • Er moet voorzichtig worden omgegaan met persoonlijke gegevens. Privacy is van groot belang. • Het moet makkelijk werken dan GroeneWielen en dat de scheiding tussen het betaalsysteem, de parkeergelegenheden en de rest van het systeem eenvoudig te maken is. Het betaalsysteem en het systeem voor de parkeergelegenheden worden t.z.t. verkocht aan de overheid. Via API, gemaakt tijdens dit project, kunnen deze systemen benaderd worden.

Tabel 5: Stakeholder Alex Aanvoerder

Naam	Eduard Erfgeest
Functie	Bank
Omschrijving	De bank financiert een deel van het project in ruil voor een platform voor de betalingen
Concerns	• De bank wil hun betalingssysteem (pay-by-fingerprint) promoten en doet alleen mee als dit betalingssysteem gebruikt wordt. • De gebruikers dienen of een rekening bij de bank te hebben of moeten hun credit card gegevens aanleveren. Tevens moeten zij de app van de bank installeren.

Tabel 6: Stakeholder Bank

Naam	Frederieke Vrijheid
Functie	Bits of Freedom representant
Omschrijving	Bits of freedom financiert een deel van het project en in ruil wil het de klantprivacy showcasen
Concerns	• Bits of Freedom is geïnteresseerd in het ontwikkelen van een standaard waar iedereen die in te toekomst in aanraking komt met Call-a-Car zich aan zou moeten houden. • Bits of Freedom wil dat persoonsgerelateerde data niet wordt gecommuniceerd met de parkeerfaciliteiten of andere partijen. • Bits of Freedom wil dat het publiek meer bewust wordt omtrent privacy en voordelen ziet in een garantie omtrent privacy.

Tabel 7: Stakeholder Bits of Freedom

2.2 Stakeholdergroep Ontwikkelaars

Deze paragraaf beschrijft de stakeholders van de groep Ontwikkelaars.

Naam	Berna Bouwer
Functie	Software Architect
Omschrijving	Zij is verantwoordelijk voor de softwarearchitectuur van het systeem.
Concerns	• De ontwikkelaars willen een state-of-the-art product maken, waardoor zij moderne technieken, methodieken en ontwikkelomgevingen kunnen gebruiken. Hiervoor zal budget beschikbaar moeten zijn. • Een interessant, wellicht mediageniek, project staat goed op curriculum vitae. • Een aantal ontwikkelaars zal langere tijd aan het project verbonden willen blijven.

Tabel 8: Stakeholder Berna Bouwer

2.3 Stakeholdergroep Gebruikers

Deze paragraaf beschrijft de stakeholders van de groep opdrachtgevers.

Naam	Corinne Castelijn
Functie	Klant
Omschrijving	De klant bestelt een auto en gaat hiermee rijden. Zij gebruikt nu GroeneWielen en is hierover niet enthousiast.
Concerns	• De auto moet makkelijk te regelen zijn; niet te lang van te voren moeten plannen. Dit moet makkelijker gaan dan bij GroeneWielen. • Voor gebruikers is het niet nodig om te investeringen in de aanschaf en onderhoud van auto's. • Gebruikers willen alleen betalen voor de kilometers die met de auto gereden worden en geen dure abonnementen waar je moeilijk vanaf kan komen. • Gebruikers hoeven geen investering te doen in het behalen van een rijbewijs, indien dit juridisch mogelijk.

Tabel 9: Stakeholder Minister van Verkeer

2.4 Stakeholder beheerders

De onderstaande stakeholders zijn te kenmerken als beheerders van het systeem.

Organisatie	Call-a-Car
Functie	Service medewerkers
Omschrijving	Deze medewerkers leveren overzichten, lossen problemen van klanten op en zijn aanspreekpunt voor derden
Concerns	• Willen op eenvoudige wijze problemen van de systemen en de klant kunnen oplossen. Hiervoor willen zij met voldoende middelen uitgerust worden.

Tabel 10: Stakeholder Beheerders

2.5 Stakeholder overig

Deze paragraaf beschrijft de overige stakeholders.

Organisatie	Provincie Limburg
Omschrijving	De provincie stelt de parkeerplaatsen te beschikking.
Concerns	• De provincie stelt de parkeerfaciliteiten ter beschikking. • De provincie wil dat de proef in Limburg plaatst vindt om de provincie als innovatief meer op de kaart te zetten.

Tabel 11: Stakeholder provincie Limburg

Naam	Autofabrikant
Omschrijving	De autofabrikant is de leverancier van de zelfrijdende elektrische auto's en zal de noodzakelijk software voor de auto's leveren a.d.h.v. de specificaties die door Call-a-Car zullen worden opgesteld.
Concerns	• Indien het systeem word geïmplementeerd, wil de autofabrikant dat zijn auto's gebruikt gaan worden. • De fabrikant wil een hogere naamsbekendheid en meer auto's verkopen. Door deel te nemen aan dit project krijgt de fabrikant een voorsprong op zijn concurrenten.

Tabel 12: Stakeholder autofabrikant

2.6 Conflicterende belangen

De onderstaande belangen botsen met elkaar:

- De overheid wil veel aan de markt overlaten, zodat de markt bepaalde diensten kan aanbieden. Dit is tegenstrijdig met het beoogde betaalsysteem, aangezien deze door de bank geleverd worden. Andere betaalsystemen zal het systeem niet ondersteunen.
- In eerste instantie worden auto's van één fabrikant toegelaten. Het gevaar bestaat dat het systeem te afhankelijk wordt van de bank en de autofabrikant, waardoor het vermarkten van het systeem moeilijker wordt.
- Indien het project een succes wordt, zullen meer mensen een auto delen. Dit heeft als gevold een lagere verkoop van auto's. De autofabrikanten kunnen dit als een bedreiging zijn.

3 Requirements

3.1 Functional requirements

De meeste functional requirements zijn eenvoudig af te leiden uit de concerns van de gebruiker en de opdrachtgever, die zeker ook als een gebruiker optreedt in de use-case beschrijving, omdat hij zich vaak verplaatst in de gedachte van de gebruiker van het systeem. Wij verwijzen dan ook graag naar de concerns om deze eenvoudige functional requirements af te leiden. Bijvoorbeeld, het is vrij duidelijk dat de gebruiker via een 1-click systeem een auto wil bestellen. Dit is een directe concern van de gebruiker. Deze concern is makkelijk te herleiden uit de use-case beschrijving. Echter, één van de requirements die hieraan ten grondslag ligt is de requirement dat het GPS systeem van de mobile telefoon van de gebruiker geactiveerd moet zijn, opdat de positie naar het deelsysteem "CaC Services" kan worden gezonden

De functional requirements waar meer aandacht aan zal worden geschonken zijn essentiële requirements die te maken hebben met het doel van het systeem. Welke requirements kunnen worden onderscheiden die tot de "core" van

het systeem zullen behoren? Dit zijn vooral de requirements die te maken hebben met de communicatie tussen de verschillende deelsystemen die reeds zijn onderkend bij het bepalen van de system boundaries.

Er zal eerst een proof-of-concept (POC) moeten worden uitgevoerd voor essentiële onderdelen van het systeem en vooral de communicatie tussen de verschillende deelsystemen. De requirements die op de volgende pagina worden genoemd zullen onderdeel zijn van de POC. De succes van het systeem zal afhankelijk zijn van deze requirements en er moet in vroegtijdig stadium duidelijkheid worden verschaft over de haalbaarheid van implementatie van deze requirements. De uitvoerbaarheid (feasibility) van de ingebruikneming van het systeem is dus afhankelijk van de mate waarin aan deze functional requirements kan worden voldaan.

Er wordt verwezen naar de bijlage A voor een overzicht van de geïdentificeerde functional requirements.

3.2 Non-functional requirements

In deze paragraaf zullen de non-functional requirements volgens het ISO25010:2011 model[6] in de vorm van kwaliteitsattributen beschreven worden. Deze zijn bepaald aan de hand van de concerns van de diverse stakeholders. Wij hebben deze in drie groepen verdeeld. De kwaliteitsattributen met de hoogste prioriteit zullen als eerste meegenomen worden in de ontwikkelproces.

De kwaliteitsattributen met de hoogste prioriteit zijn:

- Functional suitability (3.2.1)
- Security (3.2.2)
- Usability (3.2.3)
- Time-behavior (Performance efficiency) (3.2.4)
- Availability (Reliability) (3.2.5)

De kwaliteitsattributen met de medium prioriteit zijn:

- Scalability (Performance efficiency) (3.2.4)

De kwaliteitsattributen met de laagste prioriteit zijn:

- Maintainability (3.2.6)
- Deployability (3.2.7)

Bovengenoemde non-functional requirements leiden onvermijdelijk tot conflicten. Deze worden in paragraaf (3.2.8) uiteengezet.
De rationale achter de prioritering is de volgende:

De use-case is het bestellen van een zelfrijdende auto. De functional requirements die gemoeid zijn met deze criteria hebben dus de hoogste prioriteit, zonder deze requirements zal het product namelijk niet door de klant worden gebruikt, aangezien er reeds een alternatief is (GroeneWielen).

Door de opdrachtgever en door de sponsor “Bits for Freedom“ is het showcasen van de privacy van de klantgegevens als subdoel van het project gemaakt. Zonder de sponsorgelden van “Bits for Freedom“ zal het project niet uit te voeren zijn (of er moet een nieuwe sponsor worden gezocht). Ook de klant zal het zeer op prijs stellen dat op een veilige manier wordt omgesprongen met zijn gegevens.

Door de klant is aangeven dat het bestellen van een auto middels een 1-click systeem zeer belangrijk wordt ondervonden. Dit aspect wordt gedekt door de criteria “Usability“.

Er moet voor worden gezorgd dat er te allen tijde sprake is van een hoge beschikbaarheid. Het niet kunnen garanderen van een hoge beschikbaarheid kan zijn grondslag vinden in technische tekortkomingen, maar ook door te lage fysieke beschikbaarheid van bijvoorbeeld zelfrijdende auto’s. In beide situaties dient op een uniforme wijze terugkoppeling plaats te vinden naar de gebruiker toe.

Tenslotte is de response-tijd van een bestelling van een auto door de klant ook topprioriteit. De response-tijd valt onder de criteria "Time efficiency".

3.2.1 Producteigenschap Functional Suitability

De project-ontwikkelaanpak zal op een agile-manier worden aangepakt: Eerst de top-prioriteit zaken moeten worden ontwikkeld. Dit zijn de requirements die in paragraaf 3.1 zijn beschreven. Dit betekent ook dat initieel de criteria "Functional completeness" minder van toepassing zal zijn, aangezien bij een agile-aanpak incrementeel functionaliteit zal worden toegevoegd. Een agile-aanpak is nodig omdat we niet de concurrent de gelegenheid willen geven om eerder op de markt te komen met een oplossing. In ieder geval zullen de functional requirements moeten worden geïmplementeerd. Andere functional requirements, zoals het de implementatie van het deelsysteem "CaC Admin", zijn initieel van minder belang. Dit vanwege het feit dat het werk van de support medewerkers ook initieel door developers kan worden gedaan die "on-the-fly" rapporten kunnen genereren.

De criteria "Functional correctness" is wel zeer belangrijk. De functionaliteiten die worden opgeleverd, moeten functioneel correct zijn. Hierbij zijn de volgende zaken van belang:

- **NF.FS.1**: Cross-check van GPS-positie door terugkoppeling naar de klant. Indien de klant een auto bestelt zal er een optie zijn om met 1-click een auto te bestellen. Hierbij zal de GPS-positie van de telefoon van de klant worden gebruikt. De app van de klant zal het meest adequate adres opzoeken op basis van de GPS-positie. GPS-posities zijn per definitie niet exact, en om aan de criteria "Functional correctness" te kunnen voldoen is deze stap noodzakelijk. Nadat de app van de klant het adres heeft opgezocht en getoond, zal het de klant de mogelijkheid geven om binnen een bepaald tijdbestek de bestelling te annuleren. Er is voor deze methode gekozen om het 1-click mechanisme in stand te houden
- **NF.FS.2**: Nadat een auto een plek gereserveerd heeft op een parkeerplaats, zal de software van het "Parking Facilities" systeem de auto moeten detecteren. Er zullen twee manieren worden getest tijdens de POC. Enerzijds zal een proef worden uitgevoerd met enerzijds een transmitter in de auto en een sensor bij de slagboom en anderzijds zal een proef worden gedaan met een videosysteem dat het kenteken van de auto leest. Beide systemen zullen betreffende actie correct moeten kunnen uitvoeren, om te voorkomen dat onbevoegde gebruik kunnen maken van de parkeerplaats.
- **NF.FS.3**: Op het moment dat de klant de auto is binnengekomen, zal een sensor op de stoel van de klant detecteren dat de klant ook daadwerkelijk aanwezig is in de auto, daarna kan de auto pas vertrekken. Dit zal correct moeten werken en genoeg "lag" moeten worden ingebouwd opdat de auto niet vertrekt terwijl de klant bezig is met het instappen. Om dit zo veilig mogelijk te laten geschieden zal er een dubbele check moeten worden uitgevoerd: Zowel het portier moet gesloten zijn en de sensor van de stoel moet gedetecteerd hebben dat de klant zich in de auto bevindt.
- **NF.FS.4**: Er moet voor worden gezorgd dat acties op elkaar worden afgestemd en dat acties als geheel wel of niet worden doorgevoerd. Het kan niet de bedoeling zijn dat een bevestiging naar een auto wordt gestuurd, terwijl de registratie van data in het systeem ter calculatie van de kosten niet succesvol verloopt. Het systeem moet er voor zorgen dat **beide** acties wel of niet succesvol zijn (Data integriteit)

"Functional appropriateness" betekent de mate waarin de functies bijdragen aan het behalen van specifieke taken en doelen. Dit betekent dat aan alle requirements die genoemd zijn in 3.1 zullen moeten worden geïmplementeerd.

3.2.2 Producteigenschap Security

Verschillende stakeholders hebben aangegeven dat privacy zeer belangrijk te vinden. Hiervoor dient het systeem aan de kwaliteitsattribuut Confidentiality (Vertrouwelijkheid) te voldoen. Deze kwaliteitsattribuut geeft de mate aan waarin het systeem er voor zorgt dat gegevens alleen toegankelijk voor diegenen die geautoriseerd zijn. Deze kwaliteitsattribuut wordt geborgd door gebruik te maken van autorisatie en authenticatie. De beheerder van het systeem of een klant dient in te loggen om met het systeem te kunnen werken. Alle gebruikers van het systeem worden aan één of meerdere rollen gekoppeld. Per rol wordt aangegeven welke gegevens gelezen, toegevoegd, gewijzigd of verwijderd mag worden. Dit geldt ook voor externe systemen. Ook zijn worden aan een rol gekoppeld.

De kwaliteitsattribuut Integrity (integriteit) is voor het beschermen van de privacy ook van belang. Deze kwaliteitsattribuut geeft aan in welke mate een systeem ervoor zorgt dat gegevens alleen toegankelijk zijn voor diegenen die geautoriseerd zijn. De maatregelen die getroffen worden voor vertrouwelijkheid, zijn ook voor deze kwaliteitsattribuut van toepassing.

De volgende aandachtspunten kunnen worden geïdentificeerd:

- **NF.S.1**: Autorisatie en authenticatie en toepassing van accessmanagement m.b.v. rollen.
- **NF.S.2**: De communicatie van privacy gevoelige informatie moet worden geminimaliseerd.
- **NF.S.3**: De gebruiker moet de mogelijkheid worden geboden om een afweging te maken tussen gebruiksvriendelijkheid en het delen van privacy gevoelige informatie.
- **NF.S.4**: Er moet een 'dataretentieperiode' kunnen worden gedefinieerd. Deze periode geeft aan hoelang privacy gevoelige informatie wordt bewaard.
- **NF.S.5**: Indien de gebruik zijn abonnement met Call-a-Car opzegt, moeten privacy gevoelige gegevens uit het systeem worden verwijderd.
- **NF.S.6**: De gebruiker moet toestemming geven omtrent het inzien van historische informatie.
- **NF.S.7**: Privacy gevoelige informatie moet geencrypt worden opgeslagen.

3.2.3 Producteigenschap Usability

De gebruikersvriendelijkheid van andere systemen dan Call-a-Car is volgens een aantal stakeholders niet optimaal. Tijdens het ontwikkelen van Call-a-Car dient hier aandacht aan besteed te worden aangezien dit een concurrentievoordeel zal beteken.

- **NF.U.1**: Aan de criteria "Appropriateness recognisability" kan worden voldaan door de klant op zeer opzichtige wijze duidelijk te maken dat er sprake is van gebruikersgemak: De 1-click optie zal duidelijk moeten worden weergegeven door een grote button op het scherm te plaatsen waarmee de klant een auto kan bestellen.
- **NF.U.2**: De klant kan ervoor kiezen om persoonlijke behandeling te krijgen in de auto. Hiervoor zal de klant wel toestemming moeten geven om privacy gevoelige informatie te delen. Hier is dus een relatie met aandachtspunt NF.S.3 van non-functional requirement Security.

De criteria "learnability" is initieel van minder belang, aangezien de bediening van de functionaliteit zeer bepert en gebruiksvriendelijk is (er is maar één knop). Tijdens een tweede iteratie, waar het deelsysteem "CaC Admin" zal worden geïmplementeerd zal er aandacht moeten worden besteed aan dit aspect, opdat de gebruiker op een efficiënte manier kan worden getraind in het gebruik van de applicatie door bijvoorbeeld een "guided-tour" optie binnen de applicatie.

3.2.4 Producteigenschap Performance Efficiency

De subproducteigenschap "Scalability" kan worden geplaatst onder de producteigenschap "Performance efficiency", aangezien een applicatie die slecht is geschaald, problemen zal ondervinden met de performance, omdat de responseen of doorlooptijd van aanvragen negatief zullen worden beïnvloed naarmate er meer gebruik wordt gemaakt van het systeem. De volgende aandachtspunten kunnen worden geïdentificeerd:

- **NF.P.1** Er moet dus gebruik worden gemaakt van een technische oplossing die het toestaat om dynamisch extra resources toe te voegen indien dit nodig blijkt te zijn.
- **NF.P.2** De technische oplossing moet het mogelijk maken om meerdere request tegelijkertijd te kunnen verwerken. Aangezien er meerdere klanten tegelijkertijd een bestelling kunnen doen, en er ook meedere auto's in het spel zijn, is het belangrijk dat de oplossing die gekozen wordt, deze requirement ondersteunt.

"Scalablity" heeft niet de hoogste prioriteit, aangezien er alleen problemen met de schaling van de applicatie worden verwacht indien particulieren in de toekomst hun auto's ter beschikking kunnen stellen aan Call-a-Car. Volgens de use-case beschrijving moet hier rekening mee worden gehouden bij de opzet van de architectuur. Dit betekent dat de eerste versies van het systeem niet 100% schaalbaar hoeven te zijn.

De criteria "Time-behavior" heeft betrekking op de response- en doorlooptijd van het systeem, waarbij de volgende opmerkingen kunnen worden gemaakt m.b.t. de response-tijd van de deelsystemen:

- **NF.P.3** Indien een klant een auto bestelt, moet de app binnen drie seconden een reactie kunnen geven. De reactie zal bestaan uit een indicatie over wanneer de auto kan worden verwacht. Dit is afhankelijk van het aantal zelfrijdende auto's die door Call-a-Car zijn aangeschaft. Bij het bestellen van een auto zijn drie deelsystemen betrokken: De app, het deelsysteem "CaC Services" en de ANCS-systemen van de zelfrijdende auto's. De response-tijd van een bestelling zal dus kunnen worden verdeeld in de volgende onderdelen:
  - Communicatie tussen "CaC app" en het deelsysteem "CaC Services"
  - Communicatie tussen het deelsysteem "CaC Services" en alle beschikbare zelfrijdende auto's waarbij van deze auto's de positie zal worden opgevraagd (Divide-and-conquer)
  - Het bepalen van de meest ideale beschikbare auto (merge). Er zal een algoritme moeten worden uitgevoerd dat de kortste afstand bepaald van alle beschikbare auto's. De auto met de kortste afstand zal naar de klant worden gestuurd
  - Communicatie tussen het deelsysteem "Cac Services" naar de app van de klant

De response-tijd van de aanvraag van een parkeerplaats door een zelfrijdende auto is van ondergeschikt belang (geen prioriteit), aangezien die niet belangrijk is voor de klant.

3.2.5 Producteigenschap Availability

Niet alle systemen hoeven 100% beschikbaar te zijn. Voor een support engineer is het vervelend als het CaC Admin Systeem niet in de lucht is, maar dit heeft geen negatieve directe impact op de klant. De beschikbaarheid van de app is essentieel. Indien de app niet beschikbaar bij het bestellen van een auto, is dit in feite 'nee verkopen' en kan resulteren in reputatieschade. Het zou nog erger zijn, indien een bestelling van een auto succesvol is, maar dat, door verschillende oorzaken, de auto niet 'komt opdagen'. Dit betekent dat nadat door de CaC Services een bevestiging is verzonden dat een auto naar de klant wordt gestuurd, het nog belangrijker is dat de auto ook daadwerkelijk bij de klant aankomt, en dus het 'Auto Navigatie en Controle Systeem' en de auto zelf, zeer betrouwbaar moeten zijn. Dit is een probleem aangezien zowel de auto als de software van de auto een externe afhankelijkheid is, aangezien beide aangeleverd worden door de auto fabrikant.

Een ander probleem dat zich voordoet is het feit dat de CaC App gehost wordt door een mobiele telefoon, waar een operating systeem van Apple of Android op draait. Een lagere beschikbaarheid t.g.v. een fout in het operating systeem is dus ook een externe afhankelijkheid.

Buiten deze externe afhankelijkheden, kan wel aandacht worden besteed aan verhogen van de beschikbaarheid middels de volgende aandachtspunten:

- **NF.A.1**: CaC Services: Er dient een hot-standby oplossing te komen, waarbij in het geval van een fout in een server, een andere server te taken (tijdelijk) waarneemt
- **NF.A.2**: Voldoende beschikbare auto's. 'Nee verkopen' kan worden voorkomen door over voldoende beschikbare auto te kunnen beschikken.

Zoals reeds is vermeld, indien een bestelde auto niet kan worden gestuurd naar de klant, zal dit op uniforme wijze moeten worden gecommuniceerd naar de klant.

3.2.6 Producteigenschap Maintainability

Het project zal via een agile aanpak worden uitgevoerd. Bedrijven zoals GroeneWielen zitten niet stil en zijn waarschijnlijk ook bezig om hun systeem te verbeteren. Afhankelijk van de producten die de concurrentie op de markt zet, moet Call-a-Car om kunnen gaan met veranderingen in het ontwikkelde product. Om goed te kunnen anticiperen op deze veranderingen is het nodig dat de code op zulk een manier is ontwikkeld, dat het makkelijk is om wijzigingen door te voeren: de architectuur moet flexibel zijn en design patterns moeten worden toegepast zodat toekomstige ontwikkelaars makkelijk hun weg kunnen vinden indien zij bij het project betrokken worden. In een tijd waar goede ontwikkelaars moeilijk te vinden zijn, is dit van groot belang. De volgende zaken worden daarom benadrukt om aan de criteria's "Modularity", "Reusability", "Modifiability" en "Testability" te voldoen:

- **NF.M.1**: Er moet voor worden gezorgd dat er limieten worden gesteld aan de cyclomatische complexiteit, totale grootte van het systeem, duplicatie en grootte van de units (methodes) [10] [5]. Dit wordt gerealiseerd door gebruik te maken van sonarqube[1].
- **NF.M.2**: Ontwikkelaars moeten kennis hebben van Java Enterprise software, Spring, Dependency Injection [12] en kennis van OO design patterns [3], aangezien toepassing van deze technologieën en methodologieën leidt tot de bouw van onderhoudbare software. Tijdens het beoordelen van de curriculum vitaes en de sollicitatiegesprekken zal dit aan de orde komen.
- **NF.M.3**: Test Driven Development zal worden toegepast[7].

3.2.7 Producteigenschap Deployability

Dit project zal op een agile manier werken. Hiervoor is van belang dat de software op frequente en betrouwbare wijze op de productieomgeving uitgeleverd kan worden. De producteigenschap Deployability[2] is geen onderdeel van de ISO-25010:2011 specificatie maar is op een agile manier van werken wel van belang. De volgende eisen worden gesteld:

- **NF.D.1**: Alle software wordt binnen 10 minuten gebouw m.b.v. het toepassen van CI/CD (Continuous Integration & Continuous Deployment).
- **NF.D.2**: Alle unit testen worden binnen 10 minuten uitgevoerd. Deze eis zorgt ervoor dat na het inchecken van code het vrij snel duidelijk is of bepaalde functionaliteit omgevallen is.
- **NF.D.3**: Voor het installeren van de software op de diverse omgevingen (ontwikkel, test en productie) moet weinig handmatige acties nodig. TeamCity[2] kan dit proces ondersteunen.

Indien bovenstaande eisen op redelijkerwijze niet gehaald kan worden, zal bekeken moeten of het ontwikkelproces anders ingericht moet worden.

3.2.8 Conflicterende criteria's

Door alle non-functional requirements tegen al elkaar af te zetten, komen een een aantal conflicterende non-functional requirements naar boven. Deze zijn:

- Het systeem moet binnen 3 seconden (NF.P.3) reageren, maar moet ook rekening houden met security (NF.S.7).Het versleutelen van data kost extra rekenkracht, dus meer tijd. Tijdens het ontwikkelen zal aandacht geschonken worden aan de wijze van beveiligen. Verschillende methoden zullen met elkaar vergeleken worden.
- De klant wil privacy (NF.S.3), maar vaak ook persoonlijk benadering (NF.U.2). Het systeem zal de gebruiker moeten kunnen laten aangeven wat hij wil: privacy of een meer persoonlijke benadering.

3.3 Constraints

Deze paragraaf beschrijft de beperking die het ontwerp en de realisatie van het systeem negatief kunnen beïnvloeden. De volgende beperkingen worden, in willekeurige volgorde, onderkend:

[1] https://www.sonarqube.org
[2] https://www.jetbrains.com/teamcity

1	Het totale budget van het project bedraagt 2.000.000. Dit budget beperkt het aantal uren die besteed kunnen worden, met als gevolg dat een beperkt aantal requirements geïmplementeerd kunnen worden.
2	Het deelbudget dat besteed kan worden aan development is 1.000.000
3	Voor de afhandeling van de betalingen wordt het systeem pay-by-fingerprint van de bank gebruikt. De bank is sponsor van het budget en wil dit systeem promoten. Een ander betalingssysteem, die misschien beter en goedkoper is, is geen optie. De klant zal een app moeten downloaden waarmee betaaloverzichten kunnen worden opgeroepen en betalingen mee kunnen worden gedaan
4	Het systeem moet binnen twee jaar operationeel zijn.
5	De voertuigen communiceren via 4G met het systeem.
6	De benodigde software in de zelfrijdende auto's wordt door de autofabrikant geleverd
7	De auto's die worden gebruikt zullen worden geleverd door de autofabrikant.
8	De standaard voor de software van de parkeergelegenheden wordt beschikbaar gesteld aan andere bedrijven.
9	De CaC App ondersteunt in eerste instantie alleen Android (vanaf verise 4.4) en iPhone (vanaf iOS 8). Bij een succes kan wellicht Windows later toegevoegd worden.

Tabel 13: Constraints

Op dit moment is 5G in Nederland niet beschikbaar. De concern van de stakeholder Alex Aanvoerder is op dit moment niet reëel. Wij stellen voor om 4G te gebruiken als communicatiemiddel.
De platformen waarop de CaC-App en Cac Admin moeten werken is nog niet bekend en zal nog gedefinieerd moeten worden. Voor de CaC-App is het van belang dat deze draait op platformen waarop ook de betalings app kan werken.

3.4 Risico's

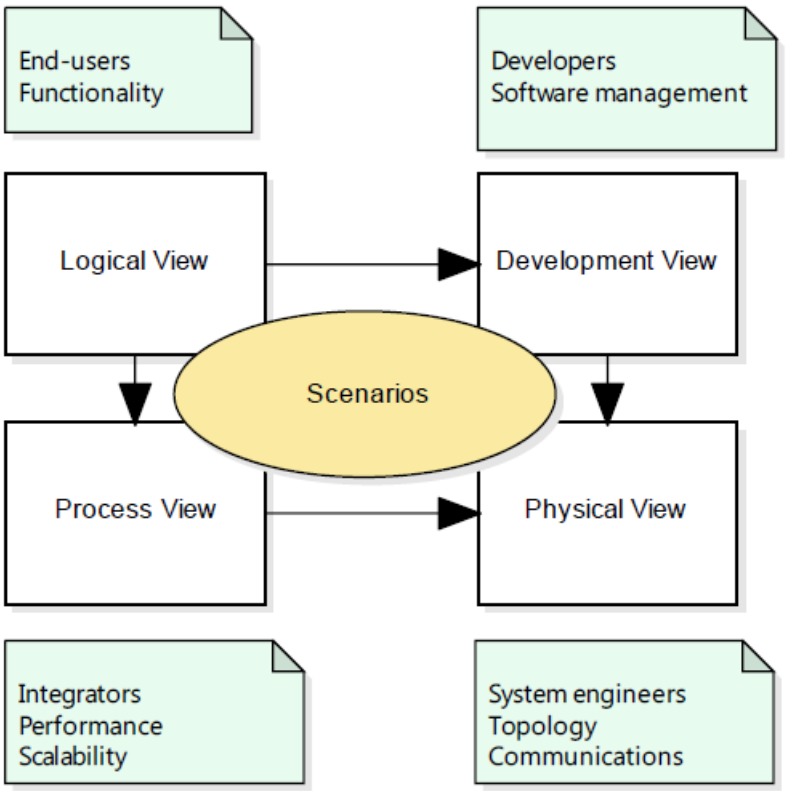
De volgende risico's kunnen worden onderkend:

- De autofabrikant zal de software leveren voor de zelfrijdende auto's. Call-a-car is dus afhankelijk van de autofabrikant betreffende de feasibility van het project.
- Er moet uitgezocht worden of auto's volgens de wetgeving zelfstandig en zonder bestuurder mag rijden. Indien dit niet toegestaan is, kan het project geen doorgang vinden.
- Developmentkosten. Er is een risico dat er meer tijd moet worden besteed aan de ontwikkeling van het systeem dan is gepland.
- Technische feasibility. Er zijn een aantal nieuwe technieken die worden toegepast en het risico bestaat dat deze moeilijk aan elkaar te knopen zijn.
- Het is op dit moment niet bekend hoeveel voertuigen de fabrikant gaat leveren en tegen welke kosten. Dit is van invloed op het aantal mogelijke gebruikers. Immers als er te weinig voertuigen zijn, moeten de gebruikers lang wachten totdat de auto voor komt rijden.
- De spreiding van de gebruikers in Nederland is nog niet bekend. Aanbeveling is om in een beperkt gebied een pilot uit te voeren, in plaats van heel Nederland. Hierdoor hoeven de auto's niet over het hele land verspreid te worden, zodat de aanrijtijden beperkt kunnen blijven.
- De provincie Limburg stelt parkeerfaciliteiten ter beschikking. Dit houdt in dat de pilot waarschijnlijk in Limburg moet plaats vinden. Limburg is minder dichtbevolkt dan de Randstad en de kans op deelnemers is daardoor kleiner. Ook is het aannemelijker dat in Limburg, vergeleken met Amsterdam, meer parkeerplaatsen beschikbaar zijn. Hierdoor is het delen van een auto minder noodzakelijk.
- De gebruikers bezitten ongetwijfeld een variëteit smartphones van verschillende leveranciers met besturingssystemen van verschillende versies. Is het mogelijk om al deze varianten te ondersteunen en is dat ook wenselijk?

4 Architecture

Wij zullen voor de beschrijving van de architectuur gebruik maken van Kruchtens "4+1"View model [8]. Dit model, zie figuur 2, bestaat uit de volgende perspectieven:

- Logische perspectief: dit perspectief biedt via class diagrammen inzicht in de relatie tussen de functionele requirements. Deze perspectief is bestemd voor de gebruikers van het systeem (4.2).
- Ontwikkelingsperspectief: een perspectief waarin de relaties tussen de verschillende software componenten van het systeem getoond worden. De ontwikkelaars gebruiken dit perspectief voor het ontwikkelen van de software. Hiervoor worden de UML component en package diagrammen gebruikt (4.4).
- Proces perspectief: dit perspectief toont het dynamische gedrag van het systeem. Hierbij zijn een aantal non-functionele requirements betrokken. Met behulp van de UML activity diagram wordt dit perspectief afgebeeld. Dit perspectief is bestemd voor de systeemontwerpers en de systeemintegrators.
- Fysieke perspectief: dit perspectief beeldt de software componenten af op fysieke componenten (zoals computers) en de relaties tussen deze fysieke componenten. In dit perspectief zijn voornamelijk de non-functionele requirements zoals scalability zijn van belang. De UML deployment diagram wordt gebruikt om dit perspectief af te beelden. De stakeholder is de systeemontwerper (4.5).
- Het overkoepelende perspectief zijn de scenario's. Vooral voor de gebruikers en voor de ontwikkelaars zijn de scenario's een hulp voor een beter begrip van de relatie tussen de functionele requirements en het systeem. De UML use case diagram wordt gebruikt om de relaties tussen de scenario's af te beelden.



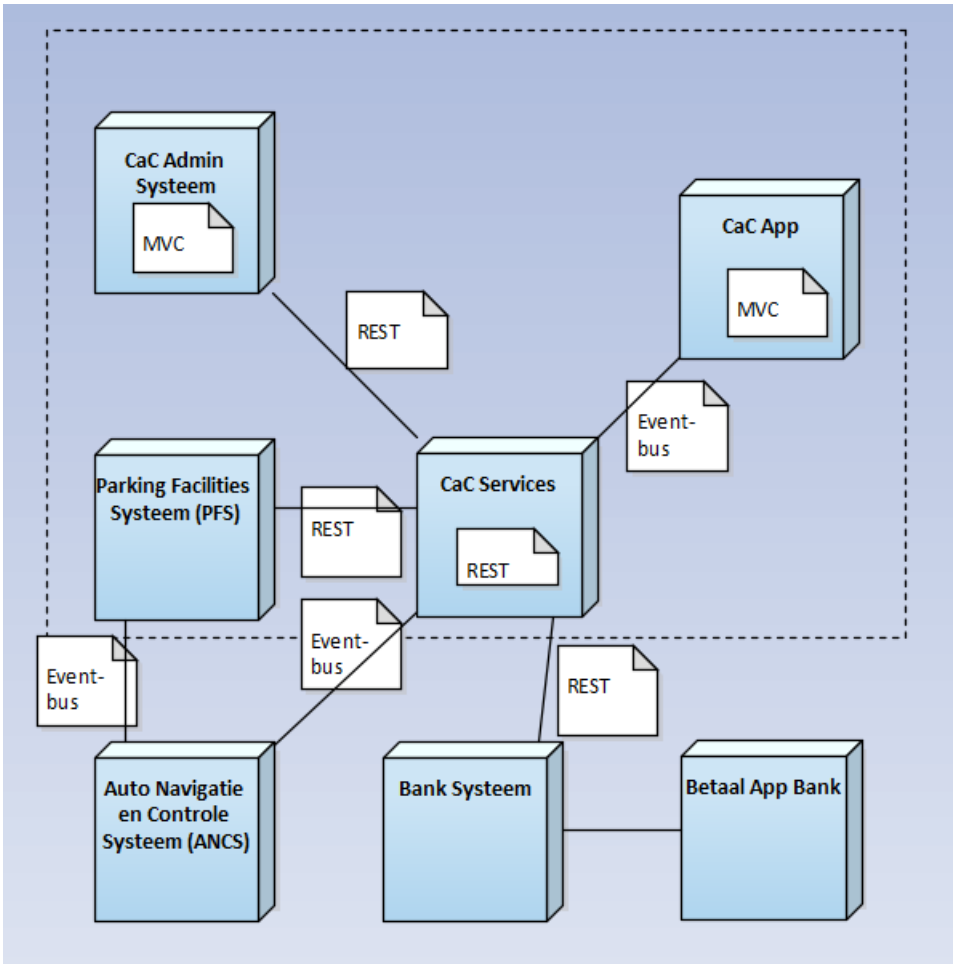
Figuur 2: "4+1"model van Kruchten

Met behulp van deze perspectieven wordt de architectuur van het systeem vanuit verschillende kanten bekeken. De stakeholders hebben elk een verschillende blik op het systeem en door de perspectieven worden deze diverse blikken bediend. De notatietaal UML 2.0[11] wordt toegepast voor het maken van de specifieke diagrammen[3].

[3]Zie `https://en.wikipedia.org/wiki/4+1_architectural_view_model`

4.1 Architectural vision

In deze paragraaf willen wij aan de hand van de belangrijkste non-functionele requirements (zie paragraaf 3.2) laten zien welke architectuur hier bij past. Voor de architectuur van het systeem zal zoveel mogelijk gebruikt gemaakt worden van bestaande en beproefde principes. Dit heeft een aantal voordelen. De architectuur is zo eenvoudig te begrijpen. De ontwikkelaars zullen hierdoor de applicatie sneller kunnen bouwen. Bovendien kan het systeem zonder veel moeite aan andere personen overgedragen worden. Dit laatste komt de criteria 'learnability' ten goede. In figuur 3 worden de



Figuur 3: Architectuurpatronen

architectuurpatronen benoemd. De architectuurpatronen die worden toegepast bij de communicatie van de verschillende deelsystemen worden hieronder beschreven:

- Event-bus architectuurpatroon. Het Event-bus architectuurpatroon zal worden toegepast bij de communicatie tussen de volgende systemen:
  - CaC App en CaC Services. Het Event-bus architectuurpatroon zorgt voor een ontkoppeling tussen twee communicerende systemen en dit is precies een requirement die van toepassing is tussen de communicatie van de Cac App en de CaC Services: Het niet beschikbaar zijn van bepaalde onderdelen van de CaC Services mag geen invloed hebben op het gebruik van de CaC App, de gebruiker moet ook in deze situatie in staat worden gesteld om een bestelling te plaatsen. Ook het terugrapporteren van de bevestiging van de bestelling dien asynchroon te gebeuren: Het kan namelijk zijn dat de telefoon van de klant tijdelijk het signaal verliest. Er zal gebruik worden gemaakt van een 'Two-way' Event-bus, waarbij een 'bestelling queue' en een 'bevestiging queue' worden gebruikt
  - De zelfrijdende auto's zullen aanvragen moeten doen naar het Parking Facilities Systeem om een parkeerplaats te reserveren. Deze communicatie zal worden gedaan met onbetrouwbare GSM-verbindingen, waarbij het

belangrijk is om deze twee systemen te ontkoppelen. Een Event-bus patroon is dus gerechtvaardigd, waarbij er ook een Two-way Event-bus principe zal worden toegepast m.b.v. twee queues

- CaC Services en ANCS. Ook de communicatie tussen CaC Services en het 'Auto Navigatie en Controle Systeem' zal berusten op het Event-bus architectuurpatroon, aangezien de communicatie tussen deze twee deelsystemen ook ontkoppeld dient te worden: Het niet beschikbaar zijn van een bepaalde auto (omdat deze zich bijvoorbeeld in een tunnel bevindt), mag niet leiden tot het verloren gaan van de aanvraag van het CaC Services naar het ANCS van de auto. Ook hier zullen er twee queues worden gebruikt, één voor de aanvraag van de positie en beschikbaarheid van de autos en één voor de bevestiging

De invloed van het Event-bus architectuurpatroon op de gestelde producteigenschappen zijn de volgende:

- Functional suitability - De functional requirements zullen vervuld moeten worden. Wat belangrijk zal zijn is de mate waarin deze vervuld kunnen worden (Functional completeness en correctness). Het Event-bus architectuurpatroon is uitermate geschikt in situaties waar verschillende systemen met elkaar gekoppeld moeten worden waar communicatie plaatsvindt over onbetrouwbare kanalen. Dit is het geval in deze use-case, aangezien zowel de communicatie tussen CaC App en CaC Services, tussen Parking Facilities System en ANCS en tussen ANCS en CaC Services zal verlopen via het onbetrouwbare mobiele 4G netwerk, waarbij communicatie is essentie niet gegarandeerd is
- Usability - Er is geen directe relatie met het Event-bus archirectuurpatroon. Echter, de gebruiker wil een gebruikersvriendelijke ervaring. Dit betekent dat de gebruiker direct feedback wil hebben nadat bijvoorbeeld een auto is besteld. Het Event-bus archirectuurpatroon biedt deze functionaliteit, aangezien de bestelling en de bevestiging van elkaar worden losgekoppeld. Binnen drie seconden wordt er naar de gebruiker gecommuniceerd of er een auto beschikbaar is, en indien het geval, wanneer de auto zal arriveren.
- Performance-efficiency/Scalability - Er zal een Event-bus technologie moeten worden gekozen, waarbij het mogelijk is om dynamisch processen toe te voegen die queue afhandeling plegen, aangezien bij toename van het aantal klanten en/of het toevoegen van meer auto's in de toekomst, er contentie zou kunnen ontstaan op deze processen en/of queue's.
- Security - Er zal een Event-bus technologie moeten worden gekozen die het mogelijk maakt om events via SSL te versturen. Tevens moet bij de communicatie tussen de verschillende systemen, er voor worden gezorgd dat er geen directe persoonsgegevens worden uitgewisseld, maar dat er gebruik gemaakt wordt van kunstmatige sleutels die als referentie kunnen worden gebruikt bij latere persoonsidentificatie
- Availability - De gekozen Event-bus implementatie zal een robuuste 'Failover' functionaliteit moeten hebben. 'Single-point-of-failure' zal ten alle tijden moeten worden vermeden. Dit betekent dat er tenminste één extra Event-bus server proces actief moet zijn

- Het REST architectuurpatroon zal in de volgende gevallen worden toegepast:
  - Het deelsysteem CaC Admin zal gegevens opvragen van de CaC Services. Laatstgenoemde services zullen nooit het initiatief nemen en in dit geval zal een REST architectuurpatroon worden toegepast. Het deelsysteem CaC Admin fungeert dus als een REST client
  - Het deelsysteem 'Parking Facilities Systeem' zal het initiatief nemen voor het versturen van gegevens naar het deelsysteem CaC Services: Het PFS weet namelijk wanneer gegevens verstuurt moeten worden, indien de auto de garage binnenkomt, wanneer de laadactie (elektriciteitsverbruik) wordt geïnitieerd en beëindigd en wanneer de auto de garage verlaat. Er is dus geen reden voor het CaC Services om een actie te initiëren: Ook hier het REST patroon worden toegepast
  - CaC Services zal periodiek een aanvraag doen naar het Bank Systeem om overzichten te verkrijgen van betalingen die zijn gedaan door de klanten. Het Bank systeem biedt een set van REST web-services (aanname) aan. Bepaalde componenten binnen CaC Services zullen dus in deze situatie als een REST client optreden
  - Verschillende business services binnen CaC Services zullen worden 'gewrapped' in REST (Server) web-services. De betreffende business services zullen o.a. de volgende verantwoordelijkheden hebben:
    * Het opslaan van gegevens die worden verzameld van het Parking Facilities systeem
    * Het berekenen van de meest ideale auto bij de bestelling van een klant
    * Het verzamelen van informatie betreffende de generatie van maandelijkse overzichten voor het CaC Admin Systeem

    De REST clients in deze gevallen zullen dus de componenten zijn die de REST web-services aanroepen:

    - Controller-componenten (MVC Context) van de CaC App en het CaC Admin System
    - Event-bus server componenten, die zich bezighouden met het ontvangen en verzenden van events, zullen de REST web-services aanroepen voor het uitvoeren van business logica, eventueel via een tussenlaag om een ontkoppeling tussen de Event-bus server componenten en de REST architectuur tot stand te brengen

  De invloed van het REST architectuurpatroon op de gestelde producteigenschappen zijn de volgende:

  - Security - Het gebruik van het REST patroon betekent per definitie dat state management zal worden uitgevoerd door de REST client. Er moet dus voor worden gezorgd dat er wordt gecommuniceerd over een beveiligde connectie middels SSL. Zoals is reeds aangegeven bij het vorige architectuurpatroon, mogen er geen persoonsgegevens worden gecommuniceerd worden tussen de REST client en de REST web-service, maar dient er gebruik gemaakt te worden van kunstmatige identifiers. De REST web-services zullen via business services communiceren met een database. Het is van belang dat gebruikersgevoelige informatie geencrypt wordt opge
  - Performance-efficiency/Scalability - Er zal een Application Server moeten worden gekozen die scaling 'out-of-the-box' ondersteunt: Er moeten dynamisch extra server processen kunnen worden opgestart indien dit nodig zou zijn
  - Availability - Net zoals bij de bespreking van het Event-bus architectuurpatroon, zal de gekozen Application Server, waar de REST web-services draaien, de mogelijkheid moeten hebben tot de configuratie van een hot-standby functionaliteit, zodat in het geval van een fatale fout in de server, de andere server die request/response afhandeling kan overnemen

- Het MVC (Model-view-controller) patroon zal worden toegepast in de volgende twee situaties:

  - De CaC app is een interactieve applicatie, die twee hoofdfunctionaliteiten zal bieden. Enerzijds stelt het de gebruiker in staat om een auto te bestellen. Anderzijds zal het ook een beperkte subset overzichten hebben die ook door het CaC Admin Systeem zal worden aangeboden. Er is dus sprake van een verschillende soorten look-and-feels (CaC Admin Systeem en CaC App). In dit geval biedt een MVC patroon dus een uitkomst. Overigens is het MVC patroon het meeste gangbare patroon als het gaat om interactieve applicaties
  - Het CaC Admin Systeem zal ook een interactieve Web-applicatie zijn. Het is logisch dat hierbij ook het MVC patroon zal worden toegepast

  Er zijn geen uitgesproken negatieve invloeden van het MVC architectuurpatroon op de gestelde producteigenschappen.

Zoals eerder beschreven zijn een aantal kwaliteitsattributen van dit project van belang. De belangrijkste zijn security (3.2.2) en performance (3.2.4). Met behulp van tactieken kunnen maatregelen genomen worden om de kwaliteitsattributen beter in de software te implementeren [4]. De tactieken voor de kwaliteitsattribuut security zijn[1]:

- Authenticate users: controleren of gebruikers zijn wie zij beweren te zijn.
- Authorize users: controleren of gebruikers de juiste rechten hebben voor het uitvoeren van bepaalde taken.
- Maintain data confidentiality: toepassen van encryptie en beveiligde verbindingen zodat ongeautoriseerde toegang voorkomen kan worden.

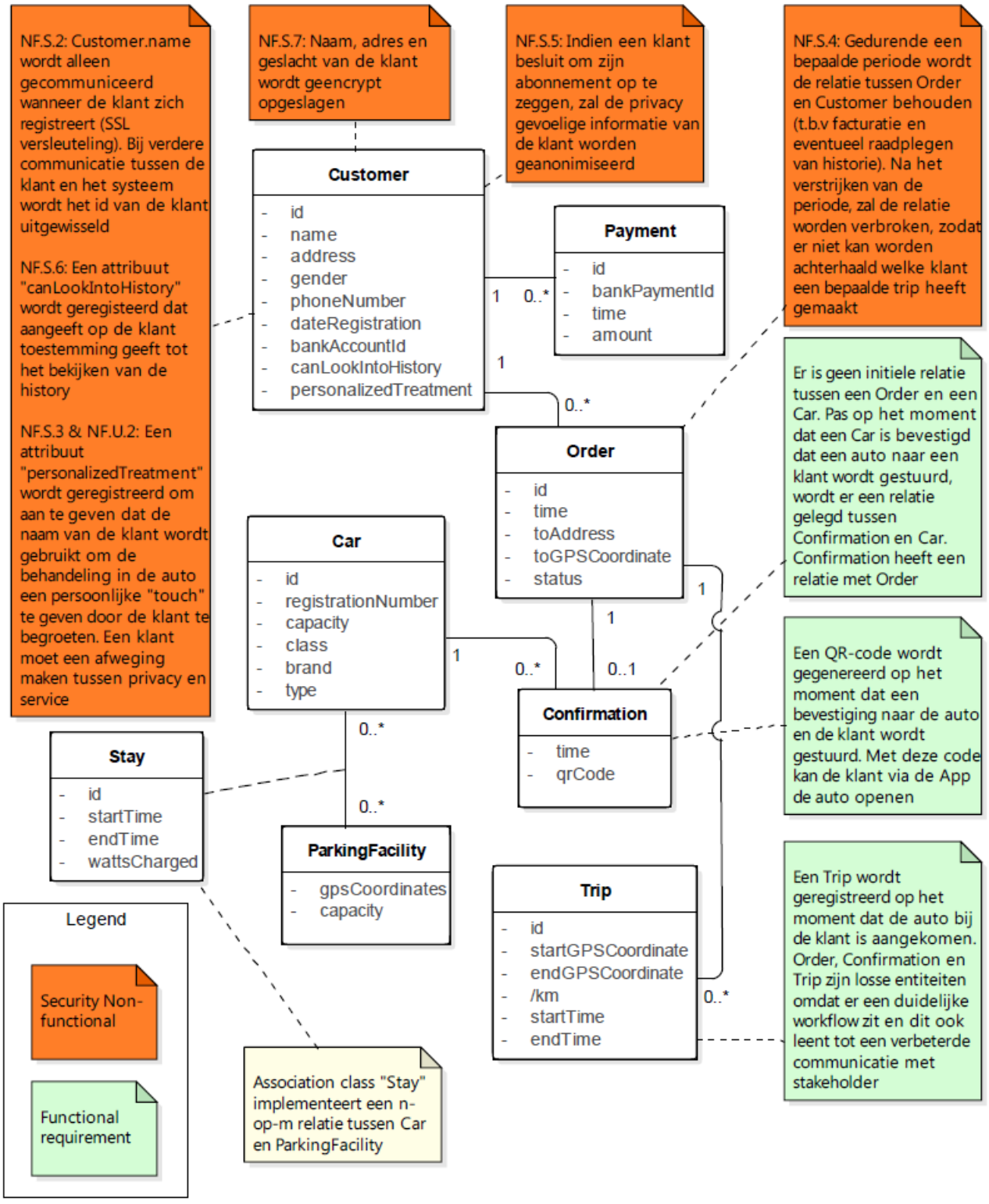
De volgende tactieken worden toegepast voor het realiseren van de kwaliteitsattribuut performance [1]:

- Introduce concurrency: door processen tegelijkertijd te laten lopen, kan meer werk verricht worden in dezelfde tijd.
- Maintain multiple copies of either data or computations: waar mogelijk wordt caching toegepast, zodat minder communicatie naar b.v. een database noodzakelijk is.
- Increase available resources: snellere en betere processoren zorgt voor snellere reactie tijden. Uitgezocht moet worden in wat binnen het budget mogelijk is.

De voorkeur gaat uit om gebruik te maken van een cloud oplossing. De teamleden hoeven dan hun tijd niet te besteden aan het inrichten en beheer van het systeem. Uitgezocht moet worden of dit inderdaad goedkoper is en dat aan Europese en Nederlandse wetgeving voldaan kan worden omtrent het opslaan van persoonsgegevens. Aangezien privacy zeer belangrijk is, zal een partij gekozen worden die hierin ervaring heeft.

4.2 Logical Viewpoint

Het Logical Viewpoint kan gebruikt worden om de functionele requirements toe te lichten. Figuur 4 betreft een UML Class diagram dat het domein model van Call-a-Car weergeeft. Naast de functionele requirements, is er ook aandacht besteed om de non-funcional requirement Security toe te lichten. Het domein model kan worden gebruikt bij de com-



Figuur 4: UML Class diagram - Domein model

municatie met de representant van de klanten om te controleren of alle entiteiten zijn onderkend en of de concepten op een juiste manier worden geïnterpreteerd. Bits for Freedom zal geïnteresseerd zijn op welke manier gegevens worden opgeslagen. Het domein model is natuurlijk geen fysiek model, maar geeft wel de mogelijkheid om te verduidelijken op welke manier en welke soort informatie gepersisteerd zal worden.

4.3 Process Viewpoint

In het process viewpoint zullen non-functional requirements zoals Scalability en Performance tegen het licht worden gehouden. Als basis voor de discussie dient figuur 5. De gekozen oplossing is een gedecentraliseerde oplossing, waarbij de keuze voor de bepaling van de ideale auto, gebaseerd op beschikbaarheid en afstand, mede door de auto zelf wordt gedaan. Op elk moment in de tijd, is alleen een zelfrijdende auto zich bewust van zijn positie. Voor de bepaling of een auto zich binnen een bepaalde (geconfigureerde afstand) van een klant bevindt, is het dus logisch deze actie ook door de auto zelf te laten nemen (Information Expert Grasp Design Pattern[9]). Er wordt dus gebruik gemaakt van de gedecentraliseerde rekenkracht van de deelsystemen ANCS die aanwezig zijn in de auto. Indien een auto bepaalt dat deze zich binnen een bepaalde afstand van een klant bevindt (en beschikbaar is), dan zal de auto een event op een queue zetten. Het deelsysteem CaC Services zal gedurende een vooraf gedefinieerde tijd, bevestiging ontvangen van zelfrijdende auto. Indien het binnen deze periode, 1 or meerdere bevestigingen ontvangt, zal het de meest ideale auto (qua afstand) selecteren uit de beschikbare auto's en een bevestiging sturen naar een topic die zowel door het ANCS van de zelfrijdende auto als de CaC App van de klant zal worden gelezen.

Een alternatief voor bovenstaande oplossing, is een gecentraliseerde oplossing. In deze gecentraliseerde oplossing, wordt door het deelsysteem CaC Services zelf bijgehouden waar de auto's zich bevinden. Om de positie van alle auto's te bepalen, zal het systeem op 'geschedulde' momenten, een aanvraag tot positie- en beschikbaarheidsbepaling aan alle ANCS'en moeten doen. Vervolgens zal door de alle ANCS'en de positie en beschikbaarheid worden doorgegeven (via queues). Indien deze oplossing wordt gekozen, zal het deelsysteem CaC Services, indien een klant een bestelling doet, sneller een geschikte auto kunnen selecteren. Echter, deze oplossing is niet schaalbaar, omdat er teveel 'onnodige berichten' door het systeem zullen vloeien. In deze oplossing is het ook nodig om de positie- en beschikbaarheidsgegevens van de auto's in geheugenstructuren op te slaan. Indien er in de toekomst wordt besloten om ook auto's van particulieren in het systeem te gebruiken, zal dit leiden tot een verzadiging van de queues en topics, waardoor de kans hoger wordt dat het systeem zal vastlopen.

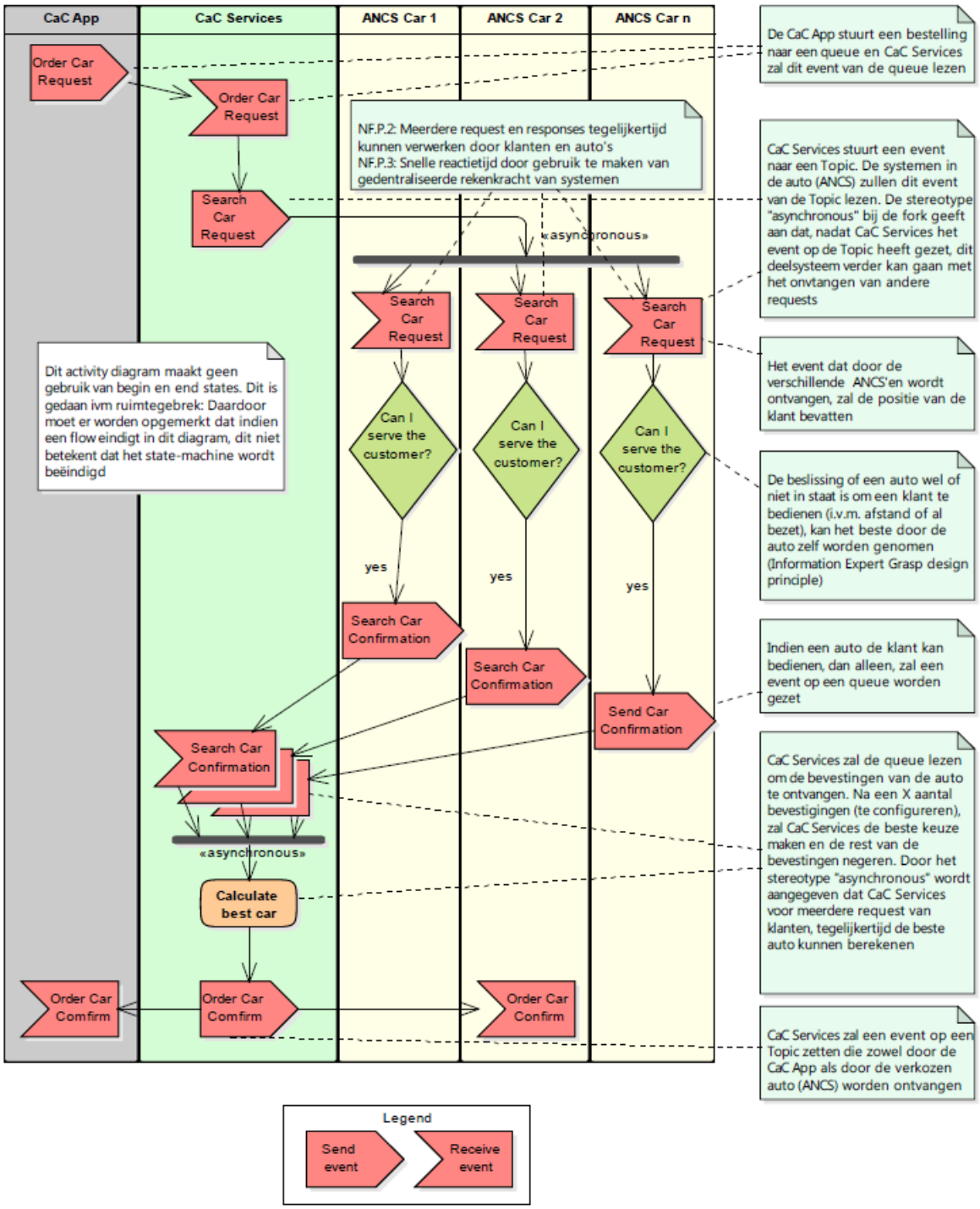
De gedecentraliseerde oplossing past het 'Divide-and-Conquer' principe toe, waarbij er geen onnodige berichten vloeien tussen de verschillende deelsystemen. Alleen indien een klant een auto bestelt, zal het deelsysteem CaC Services een request op een topic zetten met het verzoek aan de zelrijdende auto's (Divide), om zelf te bepalen (Conquer) of zij zich binnen een bepaalde afstand bevinden en of zij beschikbaar zijn. De 'merge-fase' van het 'Divide-and-Conquer' patroon zal door het deelsysteem CaC services als een manager worden uitgevoerd, door zo snel mogelijk een geschikte auto te kiezen. Opgemerkt dient te worden dat de gedecentraliseerde oplossing, bij een klein aantal beschikbare auto's, een langere reactietijd zal hebben. Echter, bij opschalen van het systeem, zal de reactietijd nagenoeg constant blijven. In het diagram wordt verwezen naar de non-functional requirements die gerelateerd zijn met 'Time-efficiency'.

Het globale Activity diagram zal voornamelijk door ontwikkelaars worden gebruikt om beeld te krijgen van de verschillende soorten berichten, events en beslissingspunten in het systeem.

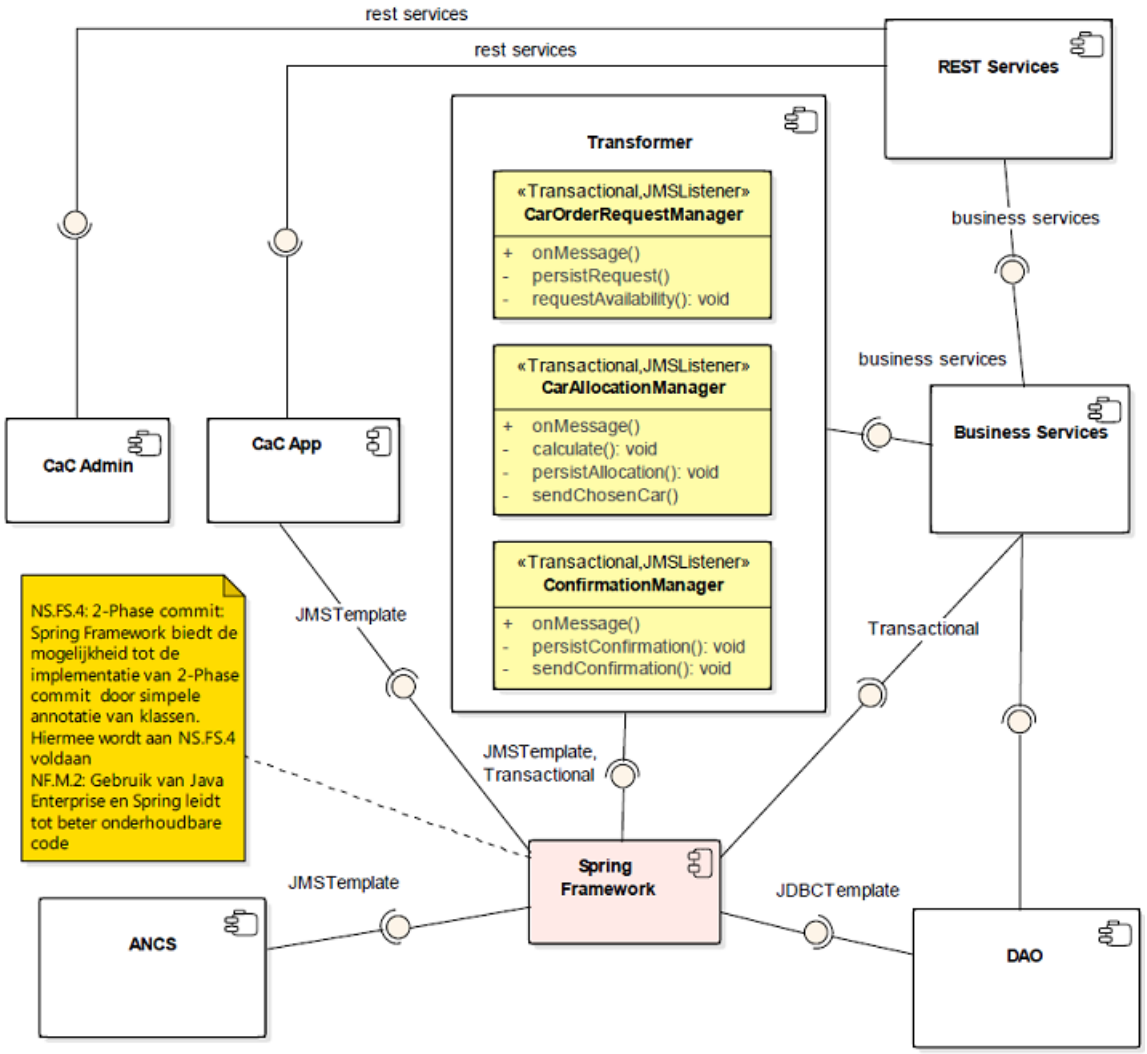
4.4 Development Viewpoint

Het diagram in figuur 6 laat zien welke componenten er zijn die een rol spelen binnen de verschillende systemen. Het Spring framework zal worden gebruikt om verschillende aspecten zoals transactionabiliteit, security en logging (cross-cutting aspecten) op een declaratieve manier te gebruiken waardoor ze dit soort aspecten niet procedureel in te code hoeven worden opgenomen (scheiden van business en non functionals)

Het component 'Transformer' zorgt ervoor dat andere componenten op een ontkoppelde manier worden betrokken bij de afhandeling van de verschillende scenario's. Een voorbeeld is de afhandeling van een verzoek van de klant om een auto te bestellen. De klasse 'CarOrderRequestManager' zal een implementatie zijn van de interface JMSTemplate. Hierdoor kan deze klasse luisteren naar berichten die zullen komen van een bepaalde queue die door het component 'CaC App' worden gevuld met bestelverzoeken. Deze klasse wordt ook getagged met een @Transaction annotatie. Mede door deze actie (er zijn nog andere configuraties nodig), zullen hierdoor alle acties binnen de onMessage-methode transactioneel



Figuur 5: UML Activity diagram - Order Car



Figuur 6: UML Component diagram

zijn [4].

De onMessage-methode van de 'CarOrderRequestManager' luistert naar inkomende verzoeken van de 'CaC App', persisteert deze verzoeken via de business services in de database (via JDBC), en zal vervolgens een broadcast doen naar alle zelfrijdende auto's door een bericht op een 'Topic' te zetten. Dit alles binnen een 2PC (Two-phase commit) transactie. Indien één van deze acties niet succesvol wordt afgesloten, worden alle voorgaande acties 'gerollbacked' en het initiële bericht van de 'CaC App' zal niet worden verwijderd, waardoor deze opnieuw zal worden aangeboden.

De voornaamste doelgroep van dit viewpoint zijn de ontwikkelaars. Zij zijn gebaat bij een componentdiagram om te bepalen welke programmaonderdelen geplaatst moeten worden in welke componenten en om welke protocollen er gebruikt worden. Ook het globale Activity diagram zal voornamelijk door ontwikkelaars worden gebruikt om beeld te krijgen van de verschillende soorten berichten, events en beslissingspunten in het systeem.

4.5 Physical Viewpoint

Het fysieke viewpoint toont met behulp van de UML deployment diagram (figuur 7) op welke nodes de verschillende onderdelen van het systeem geïnstalleerd gaan worden.Een node is een paraat en kan een smartphone, laptop, voertuig, server of de cloud zijn. De node zijn zichtbaar door het kenmerk <<device >>. Op een node kan een <<execution environement >>draaien. Voorbeelden hiervan zijn: besturingssystemen zoals Android of iOS op een smartphone, een Web browser of een Windows besturingssysteem. Specifieke technische gegevens over de cloud kan pas gegeven worden nadat een provider geselecteerd is. De deployement diagram laat ook zien welke componenten op een node geïnstalleerd worden. Door aan de non-funtional requirements categorie Deployablity (par. 3.2.7) te voldoen, zullen de diverse software componenten eenvoudig en snel gedeployed kunnen worden.

De gebruiker installeert de Cac App en de Bank App vanuit Google Playstore of Apple App Store op zijn smartphone (zie node Mobile phone). Via een internet communiceert de Cac App met de Cac Services. De Cac Services en de Parking Facilities componenten worden in de Cloud gehost. Door gebruik te maken van de Cloud is availability afgedekt, aangezien de cloud een hoge beschikbaarheid heeft. In de cloud draait een geclusterde Application Server met als resultaat een schaalbare systeem. Bij een grotere volume van het data verkeer kunnen resources snel en eenvoudig opgeschaald worden. De database draait vanwege het waarborgen van de privacy en data integriteit op een eigen server. Het beheren van een eigen server zorgt ervoor dat aan de non-functional requirement security (zie paragraaf 3.2.2) voldaan kan worden. Via een browser kan de beheerder op een laptop of een PC de CaC Admin System web applicatie benaderen. Hiervoor zijn geen apart te installeren componenten voorzien.

Een alternatieve deployment is afgebeeld in figuur 8. In dit variant is ook de database naar de cloud gebracht. Hierdoor is het voor het ontwikkelteam niet nodig om een database server te installeren en te beheren. Echter zal wel met de provider goede afspraken gemaakt (kunnen) worden omtrent privacy en data integriteit, zodat security gewaarborgd is. De non-functional requirement Availability (zie paragraaf 3.2.5)wordt door gebruik te maken van de cloud goed afgedekt.

5 Analysis

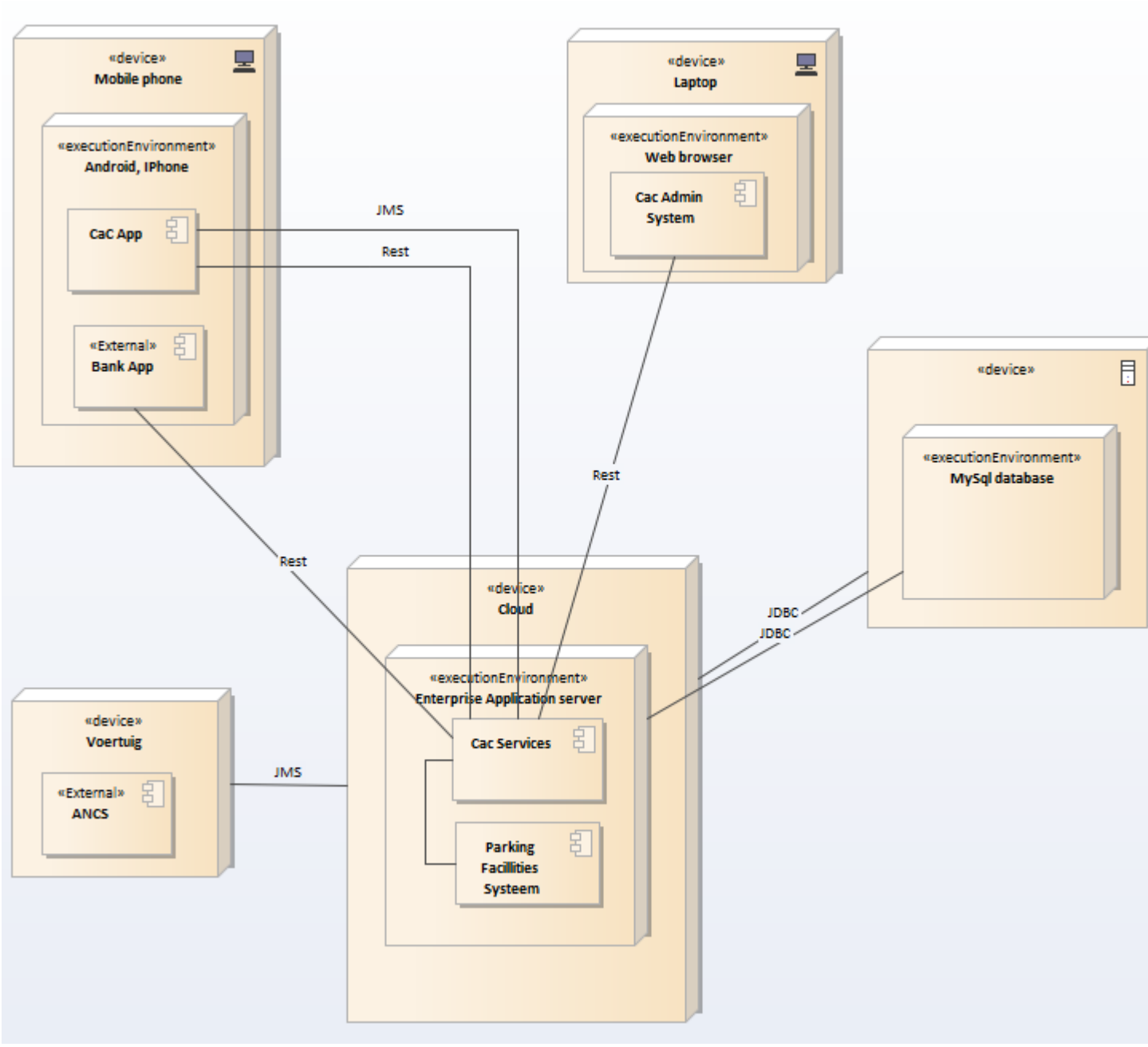
5.1 Scenario Bestel een zelfrijdende auto

5.1.1 Beschrijving

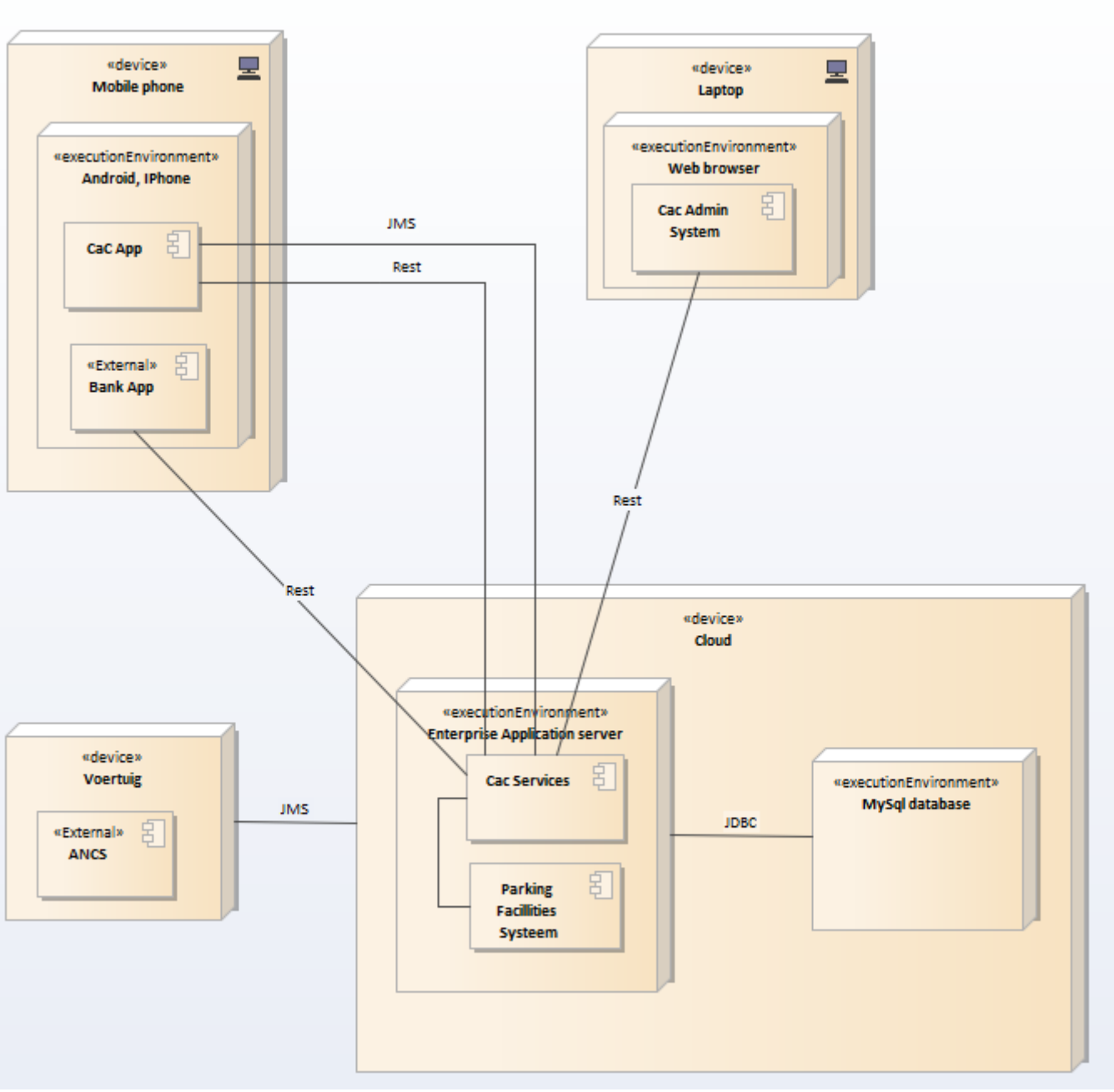
Dit scenario identificeert op zeer hoog niveau de verschillende objecten die een rol spelen bij de bestelling van een auto. Ook wordt weergegeven hoe de objecten met elkaar communiceren. Er wordt gebruikt gemaakt van een UML Communication diagram. Onderstaande punten dienen als toelichting bij diagram 5

- De klant zal de CaC App gebruiken om via een 1-click actie een zelfrijdende auto te bestellen (stap 1)
- In stap 2 wordt een message op de OrderCarRequest queue gezet. Dit betekent dat zodra het bericht is gepersisteerd, er een ontkoppeling ontstaat tussen de CaC App en de CaC services. Een eventuele tijdelijke 'hick up' in de GSM-verbinding heeft geen invloed op het verdere verloop van het scenario

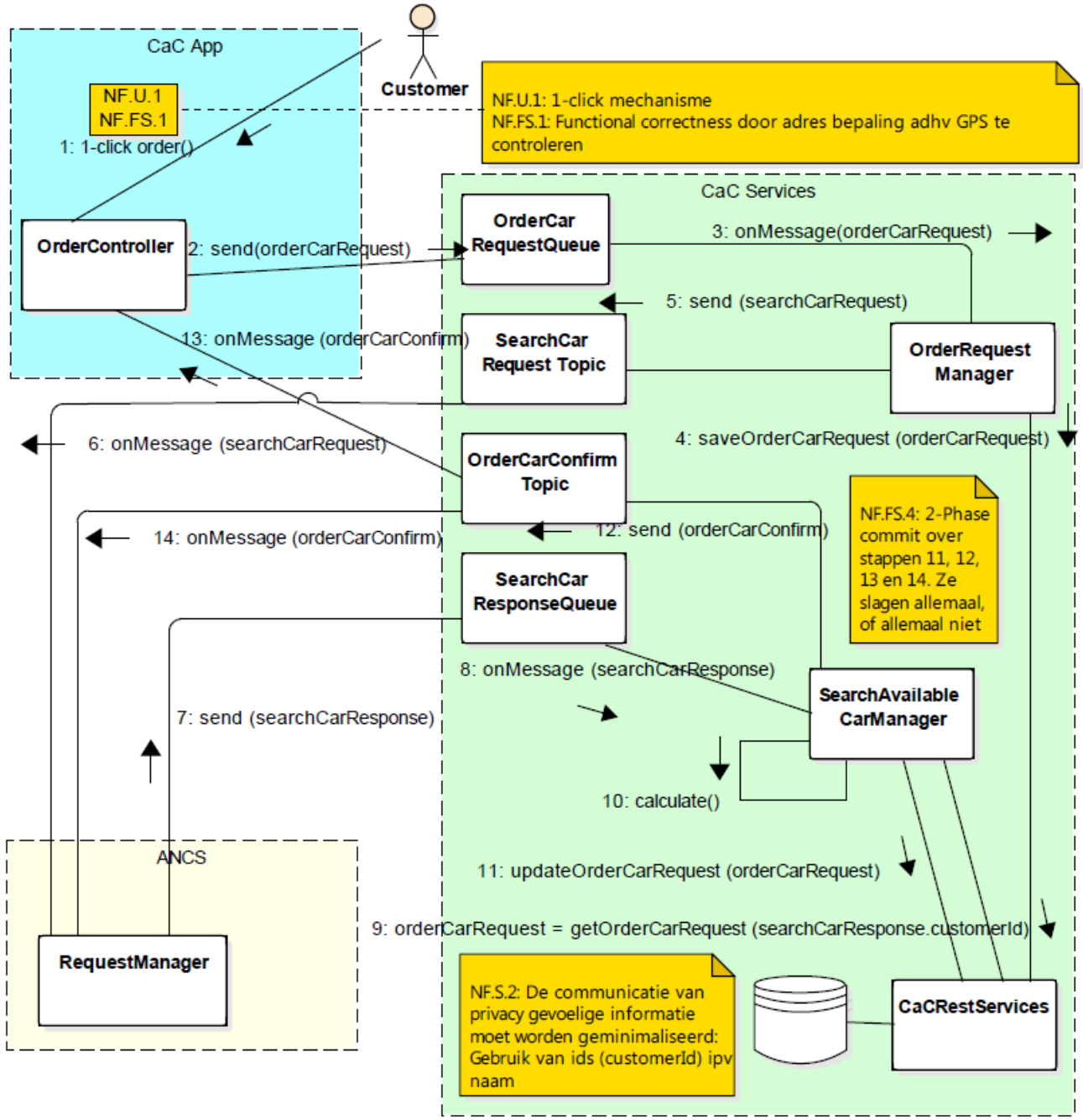
[4]https://spring.io/blog/2011/08/15/configuring-spring-and-jta-without-full-java-ee/



Figuur 7: UML Deployment diagram



Figuur 8: Alternatief UML Deployment diagram



Figuur 9: Scenario: Bestel auto

- In stap 3 wordt door een event het object orderRequestManager op de hoogte gebracht van het feit dat er een verzoek tot bestelling van een auto is gedaan. Het object zal dit verzoekobject ontvangen. Nadat dit object een verzoek tot het zoeken van een beschikbare auto op een queue heeft gezet, ontstaat er een verdere ontkoppeling. Het systeem CaC Services zou eventueel problemen kunnen ondervinden (stroomonderbreking b.v.) zonder dat

dit het scenario op een negatieve manier zal beïnvloeden

- Stap 4 zorgt ervoor dat het verzoek wordt vastgelegd in een database via een REST-webservice aanroep. Deze informatie is later nodig omdat het REST-protocol stateless is, en betreffende informatie dus moet worden vastgelegd. Dit kan niet door het OrderRequestManager zelf worden gedaan aangezien dit object alleen in het geheugen aanwezig is
- Stap 5 zorgt ervoor dat er een object op de SearchCarRequest topic wordt geplaatst
- In stap 6 zal elke zelfrijdende auto een bericht ontvangen met het verzoek om te rapporteren omtrent de beschikbaarheid van de auto en eventueel de GPS-coördinaten waar de auto zich bevindt. Deze informatie wordt op een queue geplaatst
- Elke zelfrijdende zal een bericht sturen naar de SearchCarResponseQueue, waarbij dit object de GPS-coördinaten en de beschikbaarheid van de auto zal bevatten (stap 7)
- In stap 8 wordt door het object SearchAvailableCarManager de verschillende antwoorden ontvangen van de zelfrijdende auto's
- Vervolgens wordt het verzoek van klant opgeroepen uit de database via de REST web-services m.b.v het 'customer id' van de klant (stap 9)
- In stap 10 wordt de meest ideale beschikbare zelfrijdende auto bepaald m.b.v. een korste-afstand algoritme.
- Stap 11 behelst de opslag van deze informatie om later een kostprijs te kunnen berekenen
- Er wordt vervolgens in stap 12 een bericht op de OrderCarConfirm topic gezet
- Stap 13 verduidelijkt dat de CaC App luistert naar deze queue en de informatie betreffende geschatte aankomsttijd zal worden gecommuniceerd naar de klant
- Ook de gekozen zelfrijdende auto luistert naar dezelfde topic. De auto zal zich richting het adres van de klant bewegen indien het bericht ontvangen wordt (stap 14)

De stakeholders die belang hebben bij dit scenario zijn de volgende:

- De klant - Hier wordt niet de uiteindelijke eindklant mee bedoelt, maar meer een persoon die deze eindklanten representeert en die zeer goed op de hoogte is van de eisen en wensen van de toekomstige klanten, zijnde Alex Aanvoerder
- De autofabrikant - De autofabrikant levert zowel de zelfrijdende auto's als de software (ANCS). Zij zullen namelijk deze software bouwen en deze software moet de interface gebruiken die door CaC Services wordt geleverd
- Bits for Freedom - Bits for Freedom is geïnteresseerd in hoe er wordt omgegaan met de privacy van de klanten en het scenario kan als leidraad dienen bij de bespreking van de security aspecten van de systemen
- De ontwikkelaars - Het Communication diagram is bij uitstek geschikt om inzicht te krijgen over de berichten die worden verstuurd tussen de verschillende objecten

De volgende kwaliteitseisen kunnen worden getoetst door middel van het doorlopen van het scenario:

- Functional suitability - Het scenario geeft duidelijk de functionaliteit (Functional completeness) van het scenario aan en tevens de scheiding en ontkoppeling tussen de verschillende systemen
- Functional correctness - Bij de bepaling van het adres waar de klant is, wordt er een terugkoppeling gegeven om te controleren of de verstaling van GPS-positie naar adres correct is (NF.FS.1)
- Usability - Gebruiksvriendelijkheid is belangrijk voor de klanten van Call-a-Car. Er duidelijk aangegeven dat er een 1-click mechanisme wordt toegepast aan de hand van een message dat als zodanig is gelabeld (NF.U.1)
- Security - Zoals reeds is toegelicht, kan de scenariobeschrijving worden gebruikt om als leidraad te gebruiken bij de toelichting van securityaspecten:
  - Het is duidelijk wat de communicatiekanalen zijn en dus waar informatiestromen aanwezig zijn tussen de deelsystemen en dus ook waar eventuele problemen zouden liggen

  - Met enige toelichting kan duidelijk worden gemaakt dat er geen persoonsgegevens tussen de systemen worden gecommuniceerd, maar gebruik wordt gemaakt van kunstmatige sleutels (b.v. het veld customerId in het diagram, NF.S.2)
  - Er kan worden toegelicht dat er oplossing worden gezocht om de gegevens versleuteld te versturen via een SSL-verbinding

- Availability - Dit valt niet direct uit het scenario af te leiden, maar uit het diagram valt af te leiden welke de kritische componenten zijn en dubbel uitgevoerd moeten worden om hoge beschikbaarheid te bewerkstelligen. Er wordt verwezen naar het component diagram binnen de Deployment Viewpoint.

5.1.2 Toetsing van de viewpoints

Het scenario zal worden gebruikt om de viewpoints te toetsen. Voor elk van de beschreven view points wordt verwezen naar de stappen binnen het scenario en een toelichting gegeven.

Logical Viewpoint
In het Logical Viewpoint is gebruik gemaakt van een domein model. Dit domein model geeft de verschillende entiteiten weer en de relaties tussen de entiteiten. Vooral de functionele requirements worden in het domein model belicht. Ook een non-funcitonal requirement (Security) wordt in het domein model weergegeven. Op verschillende plaatsen in het scenario wordt een verwijzing gemaakt naar requirements:

- In stap 1 wordt beschreven dat een klant de bestelling initieert. Een klant (Customer) is een entiteit binnen het domein model
- Stappen 2, 3 en 4 introduceren het concept van Order binnen het scenario. In stap 4 wordt een object van het type Order en Customer met elkaar gekoppeld en gepersisteerd in een data store
- Stappen 5, 6, en 7 helpen mee om een geschikte auto te selecteren. Een auto is een object van het type Car, zoals beschreven in het domein model
- Indien een auto is geselecteerd, wordt deze bevestigd in het systeem (Er wordt een object van het type Confirmation gecreëerd). Dit gebeurt in stap 11. Er wordt tevens een relatie gelegd tussen een object van het type Confirmation en Car
- T.b.v. de non-functional requirement ‘Security‘, aandachtpunt NF.S.2 (minimaliseren van communicatie van pricacy gevoelige informatie), wordt in het scenario (stap 9) aangegeven dat het id van de klant wordt gecommuniceerd, en niet de naam van de klant. Dit id komt overeen met het attribuut ‘id‘ van de entiteit ‘Customer‘

Proces Viewpoint
Een UML Activity diagram wordt gebruikt in het Proces Viewpoint. De non-functional requirements ‘Scalability’ en ‘Time-efficiency‘ zijn respectievelijk van medium en hoge prioriteit. Aan beide non-functional requirements wordt verwezen in dit diagram. In het Activity diagram wordt gebruik gemaakt van Send en Receive UML objecten die weergeven dat er gebruik gemaakt wordt van een Eventbus patroon. Dit wordt door het scenario bevestigd in de meeste stappen doordat in het scenario wordt aangegeven dat er gebruik wordt gemaakt van queues en topics en er zodoende een ontkoppeling ontstaat tussen de verschillende systemen

Development Viewpoint
In het Development Viewpoint wordt middels een UML Component diagram aangegeven welke software componenten er worden onderkend en via welke protocollen deze componenten met elkaar communiceren. Er wordt in het scenario middels een verwijzing naar NF.FS.4 duidelijk gemaakt dat de volgende acties onder een 2-Phase commit context vallen:

- stap 11: In deze stap wordt een ‘update‘ gedaan op de tabel Order en een ‘insert‘ gedaan in de tabel Confirmation (Datalaag)
- stap 12: Een bericht wordt op de topic OrderCarConfirmTopic gezet
- stap 13: Voorgaande bericht wordt gelezen door de afwachtende klant
- stap 14: Ook de zelfrijdende actie leest dit bericht en zal zich richting de klant begeven

Physical Viewpoint
Het Physical Viewpoint is verduidelijkt door het geven van een Deployment diagram. Het gaat hier vooral om aan te geven welke databases, application servers, mobile device kunnen worden onderkend en om de verschillende runtime omgevingen in kaart te brengen. Het is moeilijk om hier in het scenario een toelichting op te geven. Er kan worden vermeld dat indien een event wordt gestuurd naar een queue in de Application Server, deze door clustertechnologie wordt gedekt, waardoor het eventueel uitvallen van een bepaalde instance zal worden opgevangen door de andere hot-standby instance. Indien een klant dus een auto bestelt (stap 1) en en er wordt gedetecteerd dat een Application Server instance niet in de lucht is, de hot-standby instance het werk zal overnemen, waardoor de klant dus hier niets van merkt, en nog belangrijker, de workflow (stappen 2 tot en met 14) succesvol wordt afgerond, namelijk dat de klant een bevestiging ontvangt en dat de auto zich zal begeven naar het adres van de klant.

Woordenlijst

ANCS Auto Navigatie en Controle Systeem. Systeem in auto voor navigatie en controle t.b.v. de besturing van de auto..

CaC Admin System is een applicatie voor de beheerders waarmee het systeem beheerd kan worden. Er kunnen overzichten en rapportages omtrent het gebruikt gegenereerd worden..

CaC App is een applicatie waarmee de gebruiker een voertuig kan bestellen en inzage heeft in het gebruik. Applicatie wordt geïnstalleerd op een smartphone..

CaC services is het hart van het systeem en verzorgt de afhandeling van de rit aanvragen en de registratie hiervan t.b.v facturatie. Ook biedt het functionaliteit voor het opvragen van overzichten en rapportages..

Concern is een wens of behoefte van een stakeholder. Concerns worden omgezet naar requirements..

Constraint is een beperking waarbinnen het systeem gerealiseerd moet worden..

Functional requirement geeft de functies aan die het software systeem aan de gebruikers moet bieden..

GroeneWielen Een bestaand concept waarmee de deelnemers een voertuig kunnen reserveren en gebruiken. In dit geval moet de deelnemer zelf naar de auto toe gaan om deze te gebruiken. Dit product is niet gebruiksvriendelijk en is duur. .

Non-functional requirement geeft aan welke kwaliteitsattributen het software moet voldoen..

Parking Facilities System (PFS) bestaat zowel uit fysieke (slagbomen, sensoren, etc) als digitale (software) onderdelen..

Software Architectuur Een beschrijving van de componenten van het software systeem en de onderlinge interacties..

Stakeholder Een stakeholder is iedereen met een legitieme interesse in de bouw van het software systeem. Stakeholders kunnen onder andere eindgebruikers, ontwikkelaars, project management en beheerders zijn. .

System boundaries De grens tussen het systeem dat gedurende dit project ontwikkelt wordt en de buitenwereld waarmee het systeem communiceert..

Referenties

[1] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Professional, second edition, 2003.

[2] Stephany Bellomo. Architectural implications of devops, 2014.

[3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education, 1994.

[4] Neil B. Harrison and Paris Avgeriou. How do architecture patterns and tactics interact? a model and annotation. *Journal of Systems and Software*, 83(10):1735 – 1758, 2010.

[5] I. Heitlager, T. Kuipers, and J. Visser. A practical model for measuring maintainability. In *6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007)*, pages 30–39, Sept 2007.

[6] Iso/iec 25010:2011, information technology - product quality - part 1: Quality. Standard, International Organization for Standardization, Geneva, CH, 2011.

[7] Lasse Koskela. *Test Driven: TDD and Acceptance TDD for Java Developers*. Manning Publications, October 2007.

[8] Philippe Kruchten. The 4+1 view model of architecture. *IEEE Softw.*, 12(6):42–50, November 1995.

[9] Craig Larman. Applying uml and patterns: An introduction to object-oriented analysis and design and iterative development (3rd edition). pages 283–284, 2004.

[10] T. J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, Dec 1976.

[11] OMG. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, August 2011.

[12] Dhanji R. Prasanna. *Dependency Injection*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2009.

Bijlage A: Functional requirements

De volgende functional requirements kunnen worden onderscheiden:

F.1	Het systeem van de auto moet in staat zijn om de positie van de auto door te geven aan het deelsysteem “CaC Services“. Deze actie zal gebeuren op aanvraag van het deelsysteem “CaC Services“.
F.2	Indien de gebruiker via het 1-click mechanisme een auto bestelt, zal de app de GPS-positie van de telefoon bepalen en deze positie samen met de gegevens van de klant naar het deelsysteem “CaC Services“ versturen
F.3	Het deelsysteem “CaC Services“ zal alleen een auto sturen naar het adres dat gelinkt is met de GPS-positie van de aanvraag indien de gegevens van de klant bekend zijn.
F.4	Het deelsysteem “CaC Services“ moet in staat zijn om de GPS-posities en identifiers van alle beschikbare auto’s op te vragen. Een beschikbare auto is een auto die op het moment van bevraging, geen klanten naar een bestemming brengt. Op basis van het kortste-afstand algoritme zal het deelsysteem “CaC Services“ de meeste ideale auto selecteren.
F.5	Het deelsysteem “CaC Services“ moet in staat zijn via de identifier van een beschikbare auto, deze auto de opdracht te geven om naar de GPS-positie van de klant te rijden. De informatie van de klant zal aan de software van de auto worden doorgegeven. Dit is belangrijk omdat de klant met de app de auto zal openmaken.
F.6	De app van de klant moet in staat zijn om via een 1-click mechanisme de auto open te maken. Indien de gebruiker deze actie uitvoert, wordt via het deelsysteem “CaC Services“ de klantinformatie naar de software van de auto gestuurd zodat de software van de auto weet dat de auto geopend mag worden.
F.7	Indien het portier van de auto gesloten wordt en indien de sensor in de stoel detecteer dat iemand plaats heeft genomen in de auto, zal de software van de auto, het navigatieprogramma opstarten zodat de auto naar het betreffende adres zal rijden.
F.8	Indien de bestemming bereikt is, moet het software van de auto in staat zijn om de klant m.b.v. (audio)visuele features de klant op de hoogte te brengen van dit feit.
F.9	De software van de auto moet in staat zijn om te detecteren dat de klant de auto verlaten heeft. Deze actie zal worden doorgegeven aan het deelsysteem “CaC Services“ zodat kan worden berekend welke bedrag zal worden doorberekend aan de klant.
F.10	Het systeem van de auto moet in staat zijn om een reservering van een parkeerplaats te maken via het “Parking Facilities“ systeem. Daarbij zal de positie van de auto naar het “Parking Facilities“ systeem worden gezonden, en zal dit systeem de meest geschikte parkeergarage selecteren op basis van kortste afstand en beschikbaarheid van de parkeerplaatsen van de parkeergarage.

F.11	Het systeem van de auto moet in staat om na ontvangst van de geselecteerde parkeergarage en plek binnen de garage, een navigatieprogramma op te starten dat de auto naar deze locatie zal brengen.
F.12	Het deelsysteem “Parking Facilities“ moet in staat zijn om de auto te detecteren die de parkeerplek heeft gereserveerd. Dit kan gebeuren door het detecteren van een signaal die door een transmitter in de auto wordt uitgezonden, of door het lezen van het kenteken van de auto. Alleen in dien de gegevens van de auto overeenkomen met de parkeerplaats reservering door dezelfde auto, zal de slagboom van de parkeergarage worden geopend.
F.13	De software van de auto moet in staat zijn om naar de parkeerplek binnen de garage te gaan, gebruik makend van de GPS-positie van de parkeerplek.
F.14	Indien de auto is aangekomen op de plek, wordt een signaal naar het deelsysteem “CaC Services“ gestuurd, en wordt ook de beginstand van de oplaadpaal gezonden zodat met kan bijhouden hoeveel elektriciteit wordt geladen.
F.14	Indien de auto de auto een nieuwe aanvraag krijgt, moet de software van de auto in staat zijn om de nieuwe stand van de oplaadpaal te versturen naar het deelsysteem “CaC Services“ zodat kan worden geregistreerd hoeveel elektriciteit is geladen.
F.15	1 keer per maand zal het deelsysteem “CaC Services“ een facturatie uitvoeren en de verbruiksgegevens van klanten naar het bank deelsysteem sturen.

Tabel 14: Functional requirements i.v.m. feasibility / doel van het systeem