

Software Design Description

Inhoudsopgave

- Doel van dit document
 - Component Diagram
 - Overzicht van alle componenten
 - Detailed Design Description
 - UC 1 Reserveringscomponent
 - UC 2 Autocomponent
 - UC 3 Betalingscomponent
 - UC 4 Klantcomponent
 - UC 5 Autobehaercomponent
 - Design Class Diagram
-

Introductie

De opdracht staat (kort) beschreven in het SRS. Om geen duplicate informatie te verstrekken, kunt u [naar de Introductie Opdracht](#) gaan.

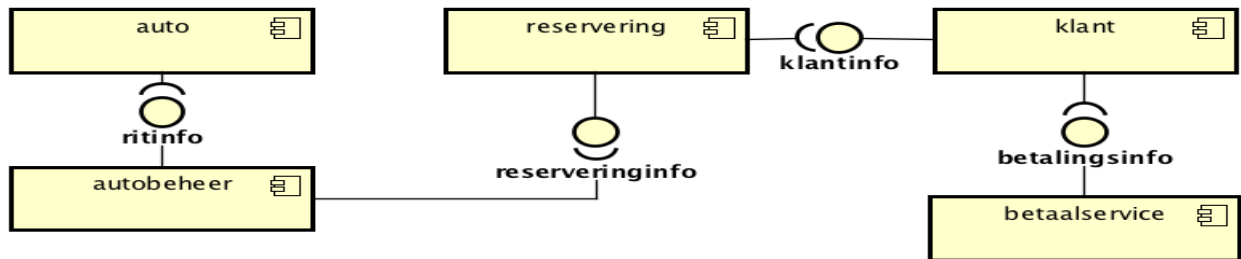
Doel van dit document

- Software Architectuur in grote lijnen beschrijven a.d.h.v. een componentdiagram.
 - Systeem operaties over de individuele componenten beschrijven a.d.h.v. Component Sequence Diagrams.
 - Een vertaling maken van concepten in het probleem(/opdracht)-domein naar code-concept a.d.h.v. een Design Class Diagram
 - 'Gedrag' van het systeem beschrijven a.d.h.v. andere interactie-diagrams zoals Activity diagrams en een statemachine.
 - Gemaakte keuzes in het ontwerp beargumenteren en verantwoorden.
-

Component Diagram

Iteratie 1

Tijdens de lessen van OSM-M hebben wij gezamenlijk een eerste versie van een component diagram opgesteld. Nadat wij deze met z'n tweeën besproken hadden, vonden het niet noodzakelijk hier tijdens de eerste iteratie nog wijzigingen in aan te brengen.

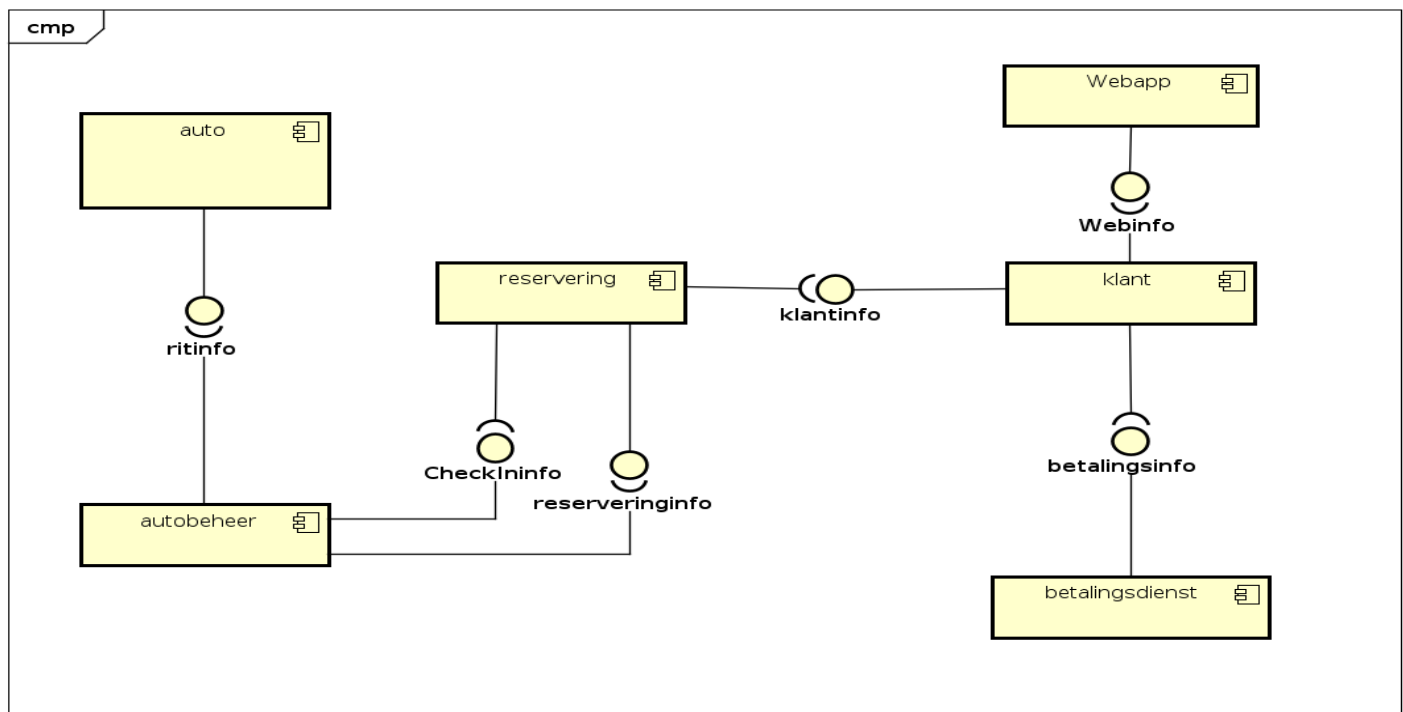


Path to image : [Component_Diagram_v1-0-1](#)

Iteratie 2

Tijdens de tweede iteratie hebben wij noodzakelijkerwijs een aantal wijzigingen doorgevoerd.

1. Een nieuwe interface: **CheckInInfo** vanuit **autobeheer** aangeboden naar **reservering**.
2. De required/provided interface omgedraaid vanuit **auto** naar **autobeheer**.
3. Een nieuw component: **Webapp** met een provided interface richting **klant**-component.
4. De naam **betaalservice** veranderd in **betalingsdienst**.



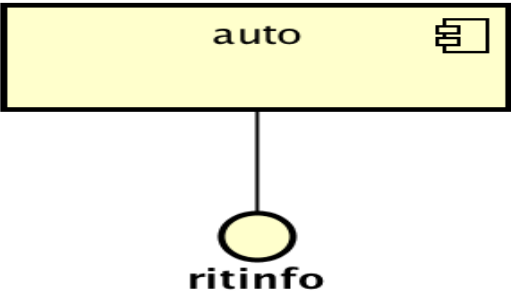
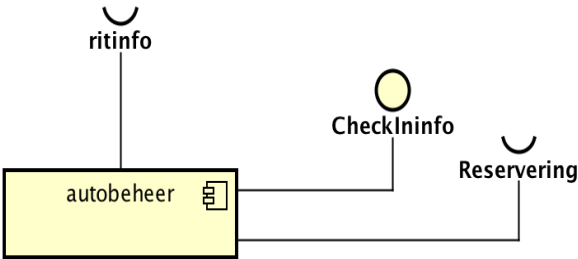
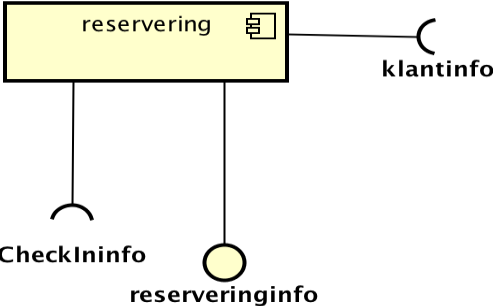
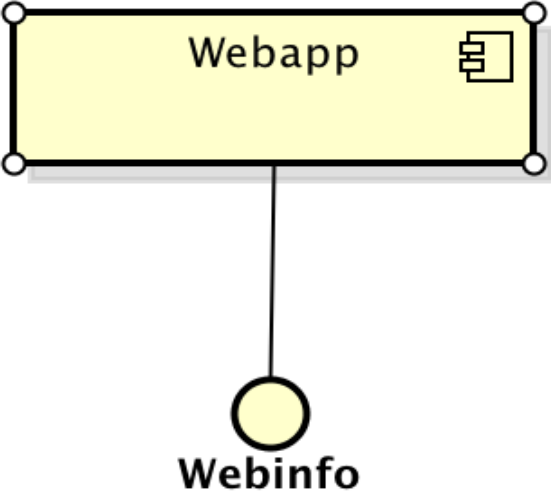
Component_Diagram_v1-0-2

Path to image : [Component_Diagram_v1-0-2](#)

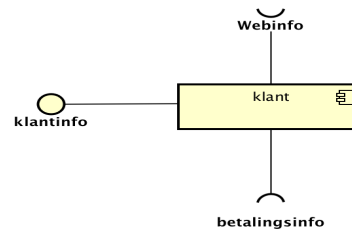
Overzicht van alle componenten

Onderstaande tabel biedt een overzicht van de provided en required interfaces per component.

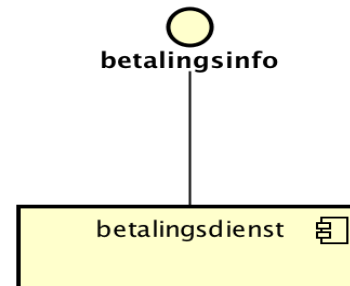
* Provided Interface
* * required Interface

Component	* Provided	** Required	Component image with interfaces
auto	RitInfo		
Autobeheer	CheckInInfo	ritInfo, reserveringsinfo	
Reservering	Reserveringsinfo	CheckInInfo, klantInfo	
webapp	webInfo		

klant klantInfo webInfo,
 betalingsInfo



**Betalings-
dienst** **betalingsInfo**



Detailed Design Description

Hieronder zijn de gedetailleerde beschrijvingen van ontwerpen van het systeem. Om de beschrijvingen structuur te geven, bespreken we de onderdelen in hoofdlijn over de Use Cases. Hierbij geldt dat het soms relevant kan zijn om uit te wijden over andere onderdelen, hierdoor ontstaat overlap tussen de use cases.

Detailed Design Descriptions Format

- * korte beschrijving van de koppeling tussen:
 - De Use Case.
 - Het ontworpen component.
- * Een component Sequence Diagram
- * Eventueel een Activity Diagram
- * Ontwerpbeslissingen met eventuele Design Patterns.

Overzicht keuze type diagram

UseCase-Code	SSD	CSD	AD	SM
UC-1	X	X		
UC-2	X	X	X	X
UC-3	X	X		
UC-4 (*CRUD)	X	X		
UC-5 (*CRUD)			X	

*CRUD = Create Read Update Delete

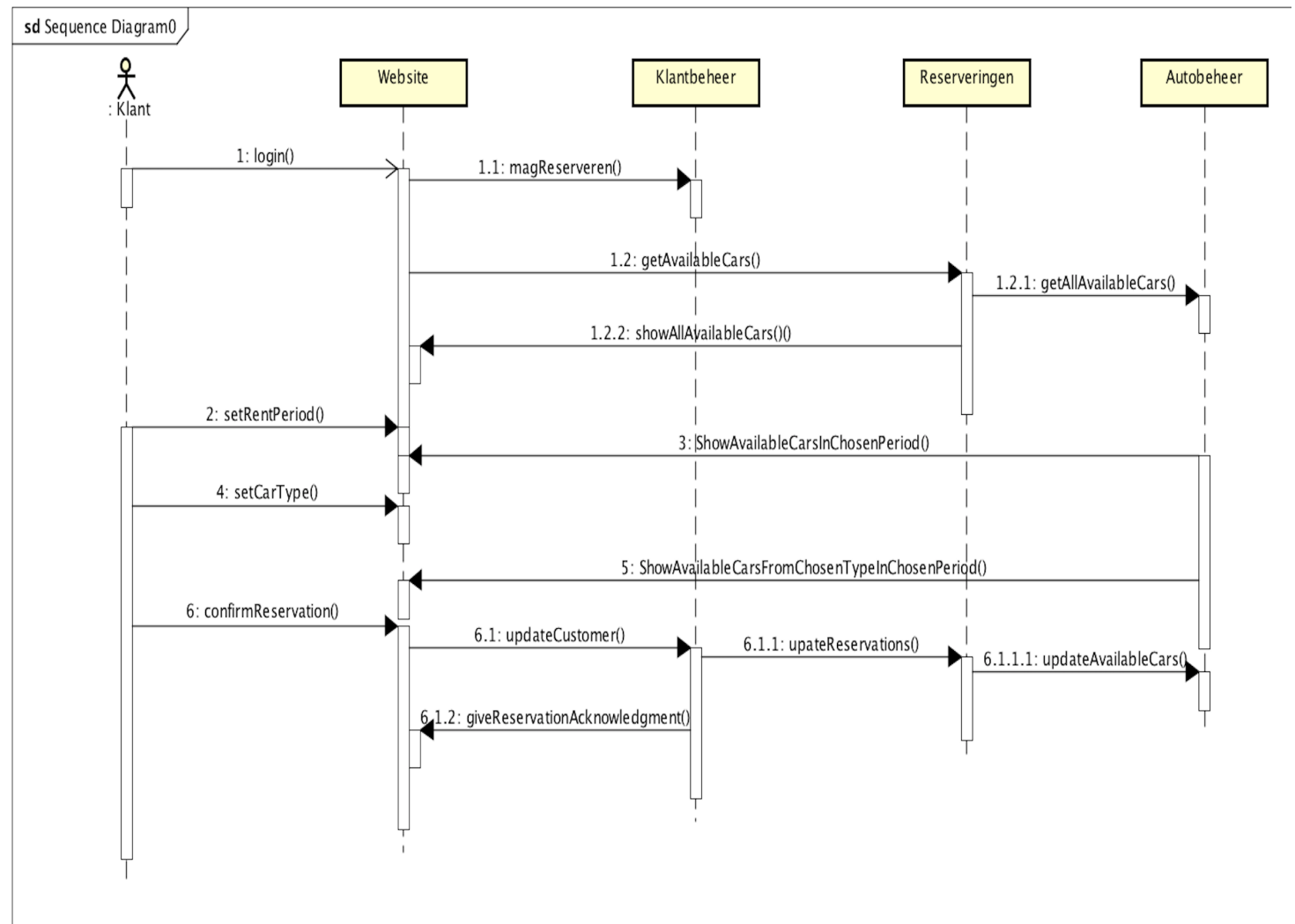
UC 1 Reserveringscomponent

Short description

Het reserveren van een auto bestaat uit grofweg 3 (hoofd-)onderdelen:

1. Inloggen en verificatie van de klant.
2. Criteria aangeven voor het huren van een huurauto
3. Bevestigen van de reservering, praktische informatie verschaffen.

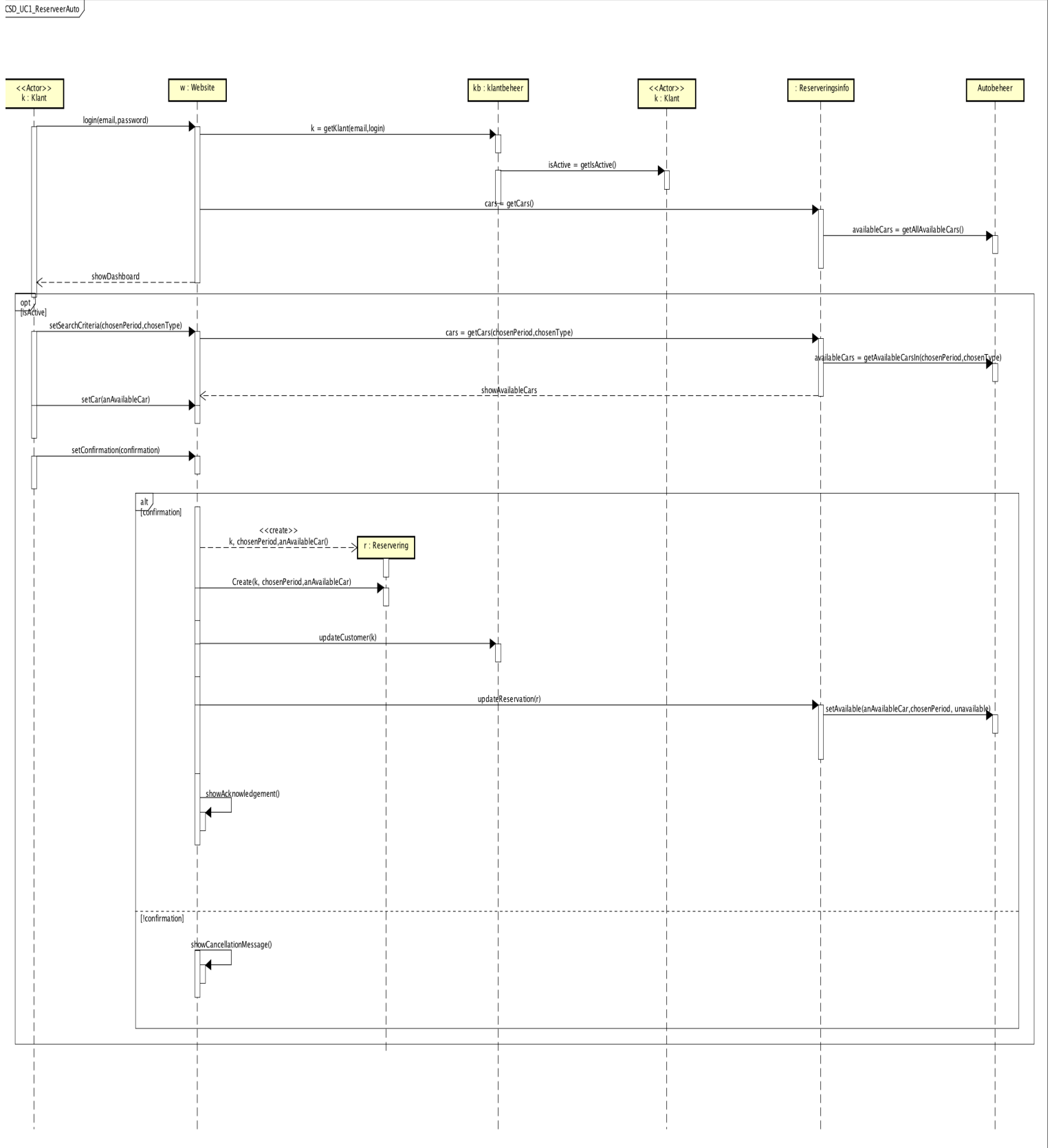
Component Sequence Diagram *Iteratie 1*



CSD_UC1_reserverenAuto_it1

Path to image : CSD_UC1_reserverenAuto_iteratie1

Component Sequence Diagram *Iteratie 2*

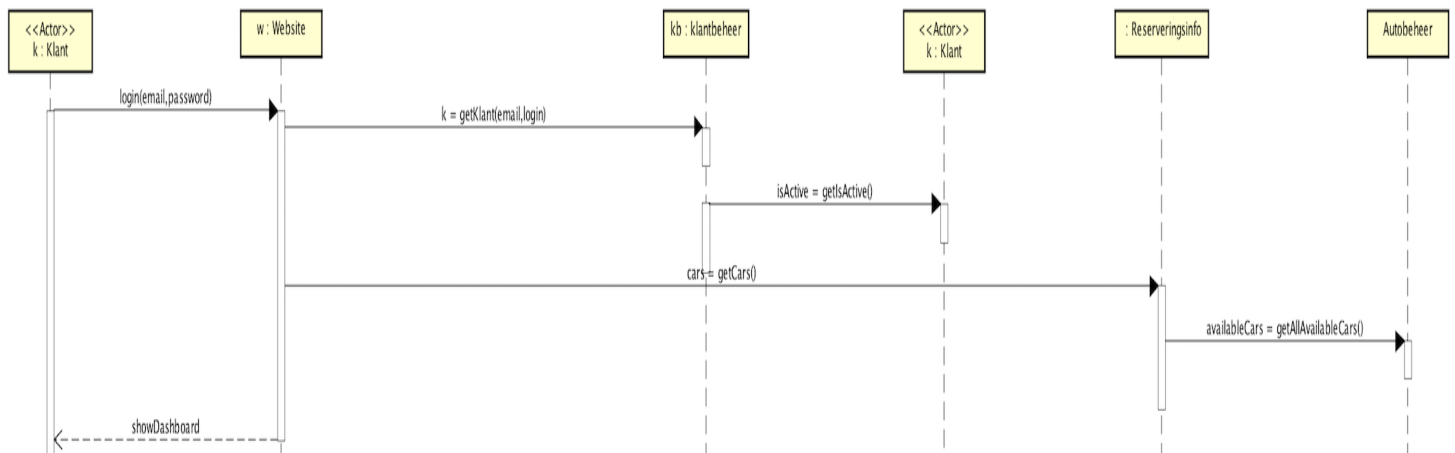


CSD_UC1_rserveerauto_it2

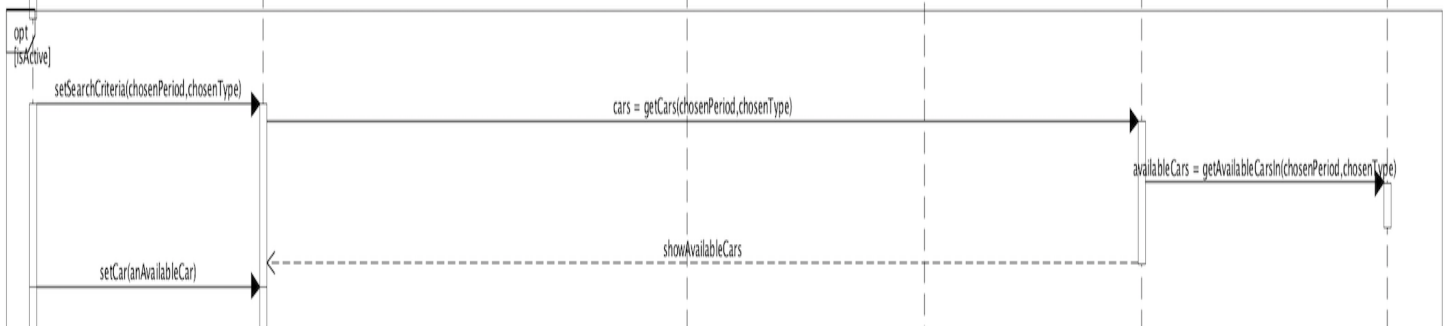
Path to image : CSD_UC1_rserveerauto_iteratie2

Design decisions

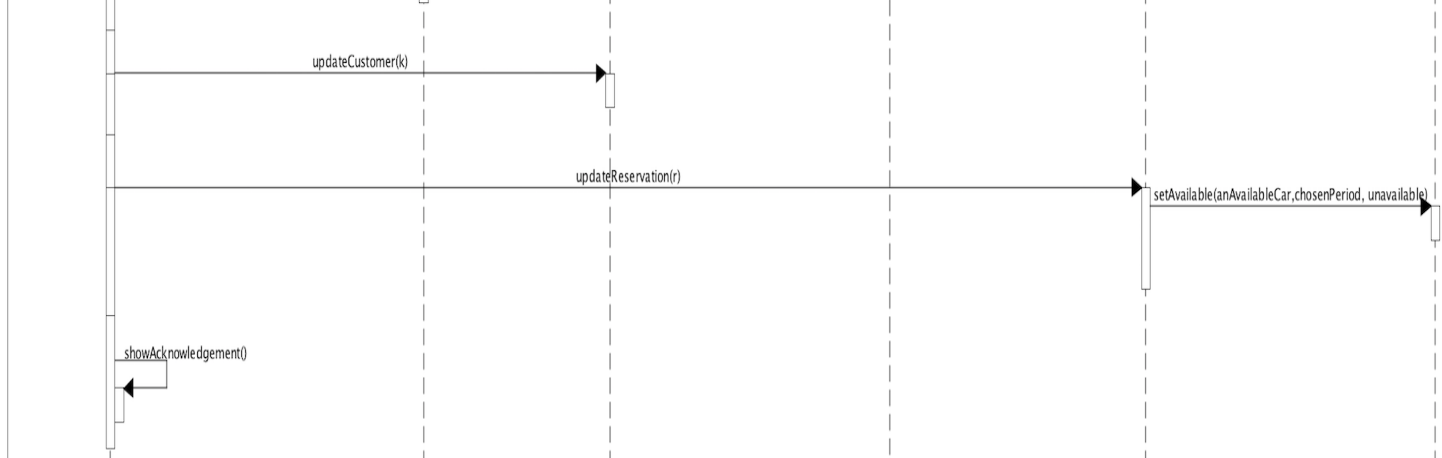
Inloggen en verificatie van de klant.



Criteria aangeven voor het huren van een huurauto



Bevestigen en praktische informatie



UC 2 Autocomponent

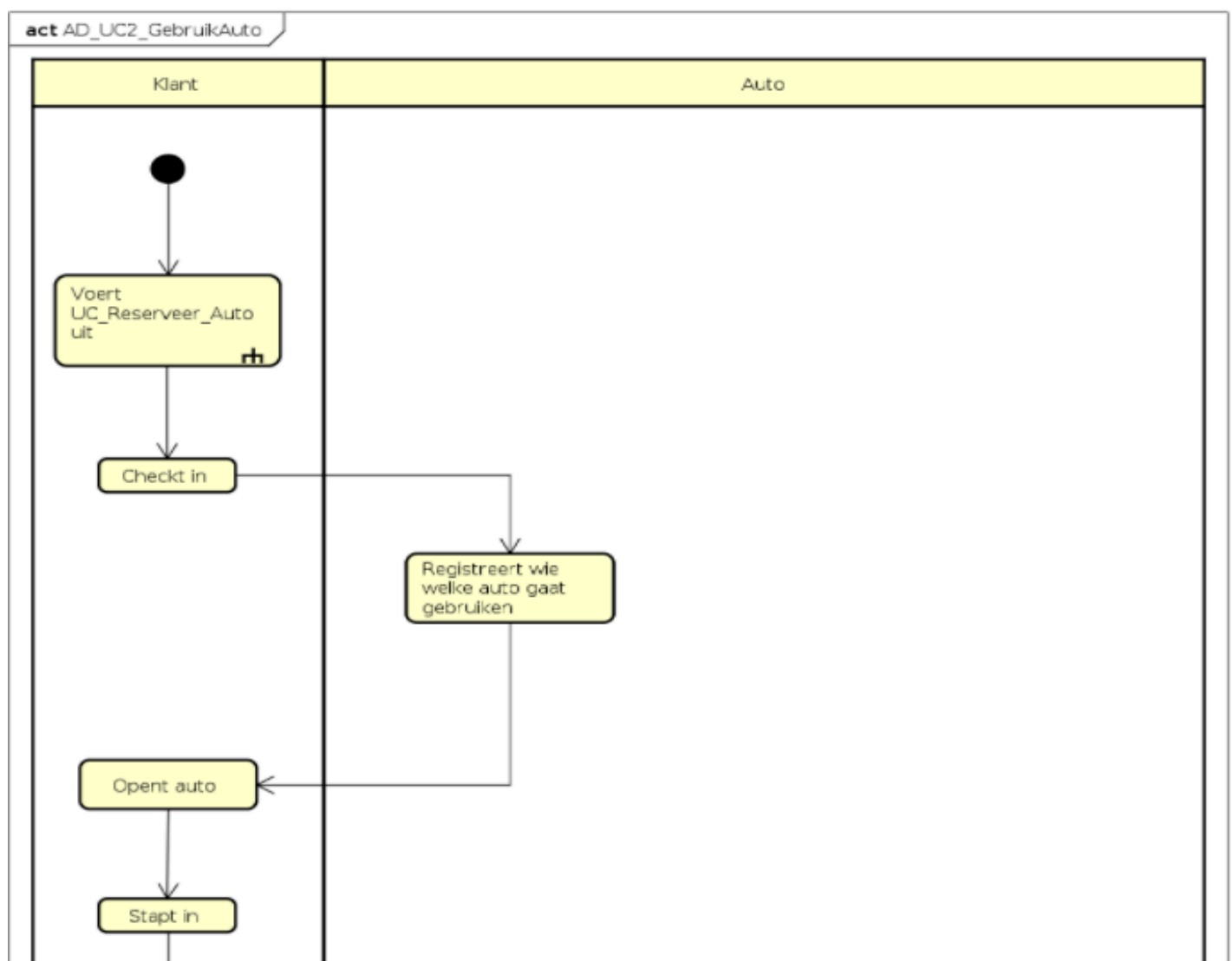
Short description

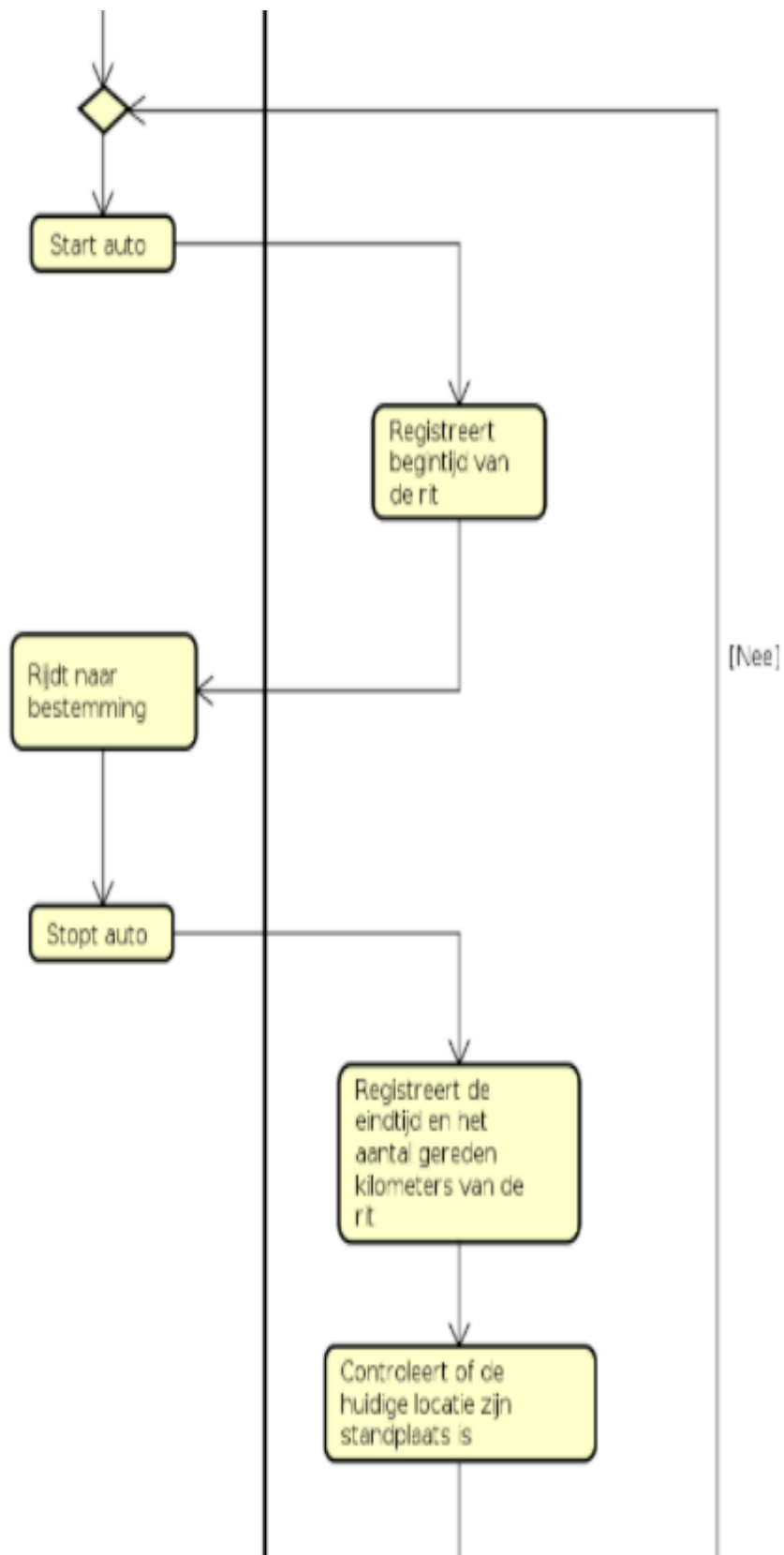
Het gebruiken van een auto bestaat 3 hoofd-onderdelen:

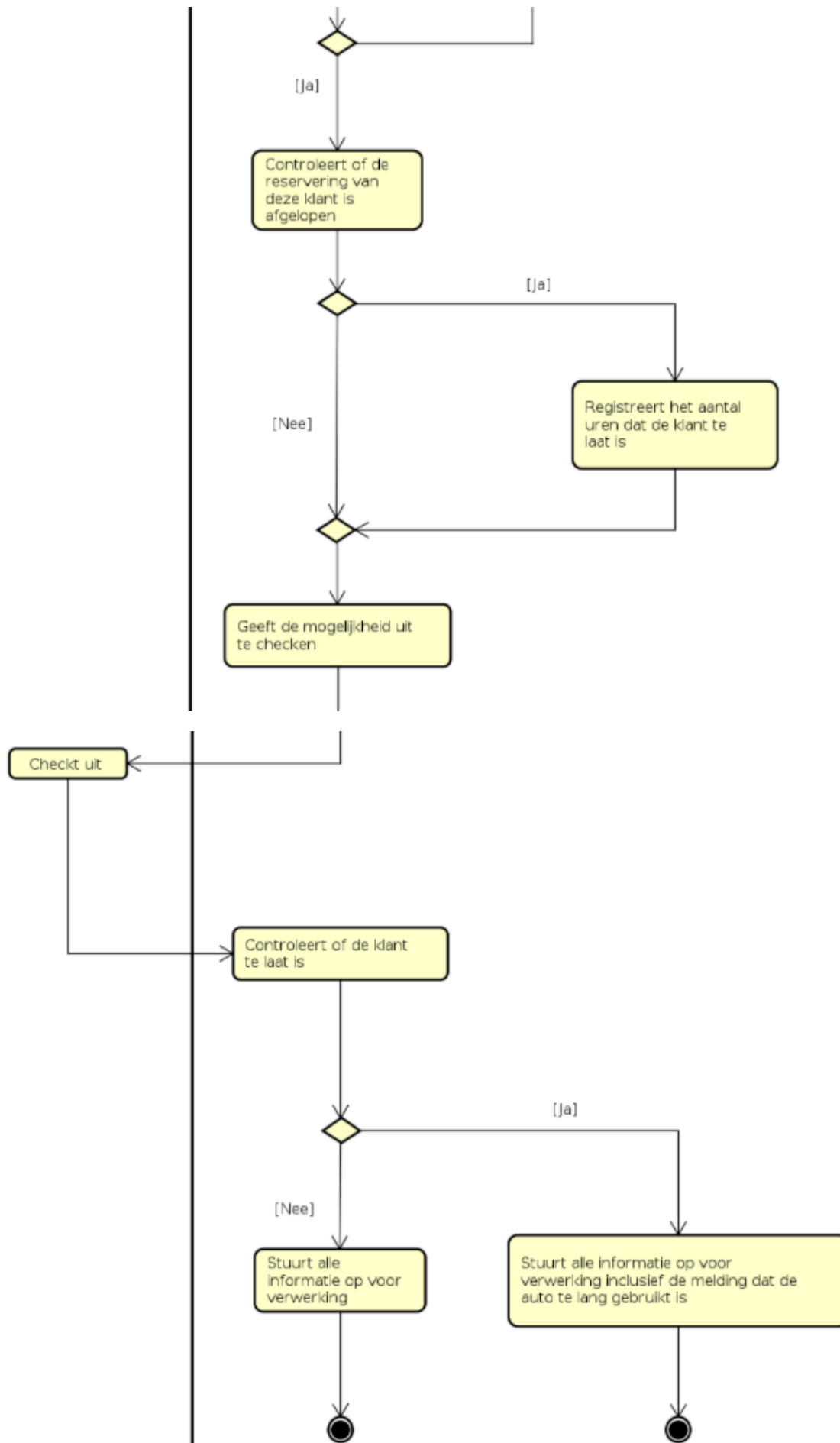
1. Openen van een auto a.d.h.v. verificatie op de reservering
2. Het maken van 1 of meer ritten tijdens de huurperiode
3. Het doorvoeren van de gemaakte ritten na de huurperiode.

Om de flow van activiteiten extra te verduidelijken hebben we voor deze Use Case ook een activity diagram gemaakt.

Activity Diagram

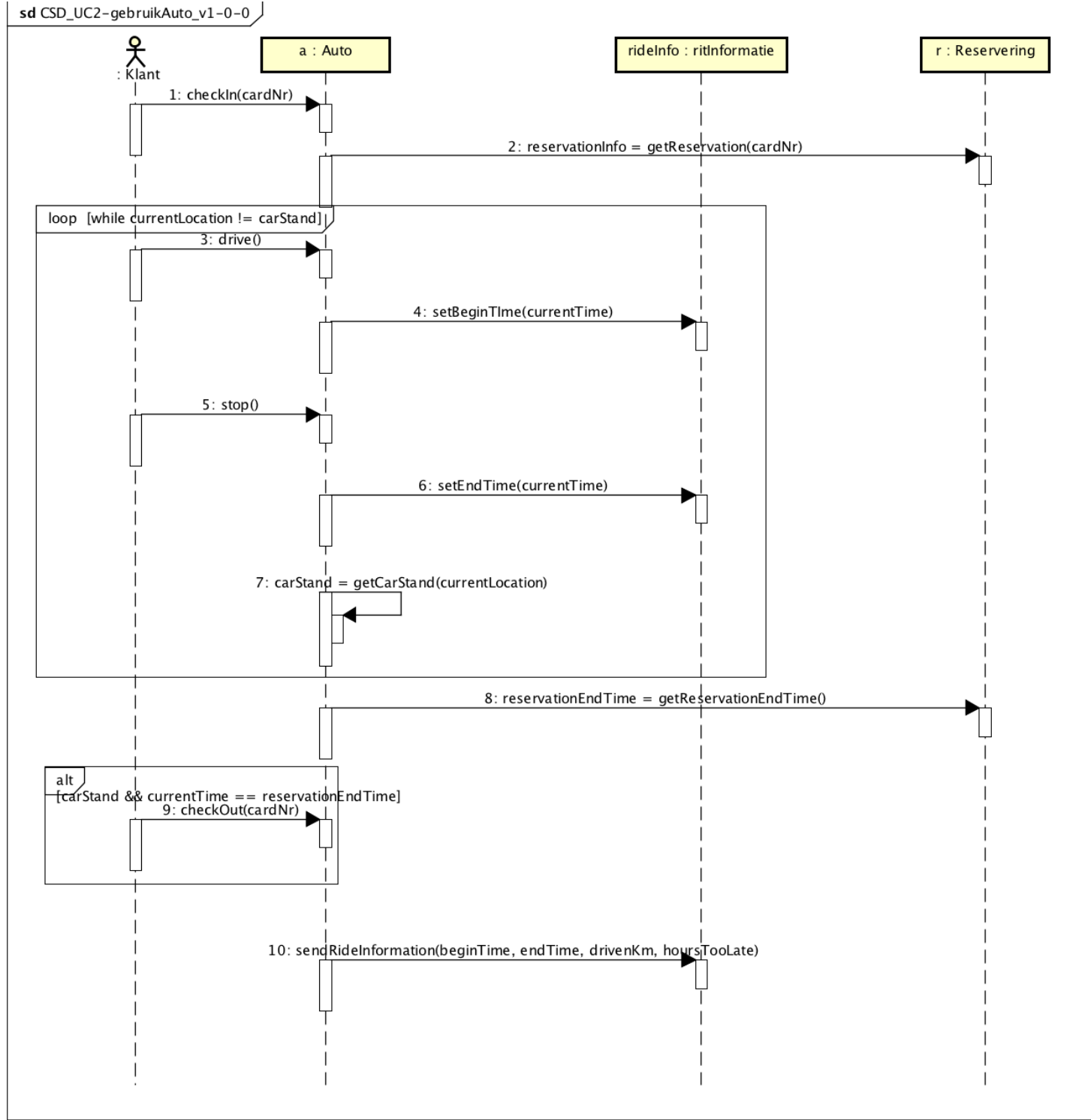






Path to image : AD_Gebruik_Auto_iteratie_2

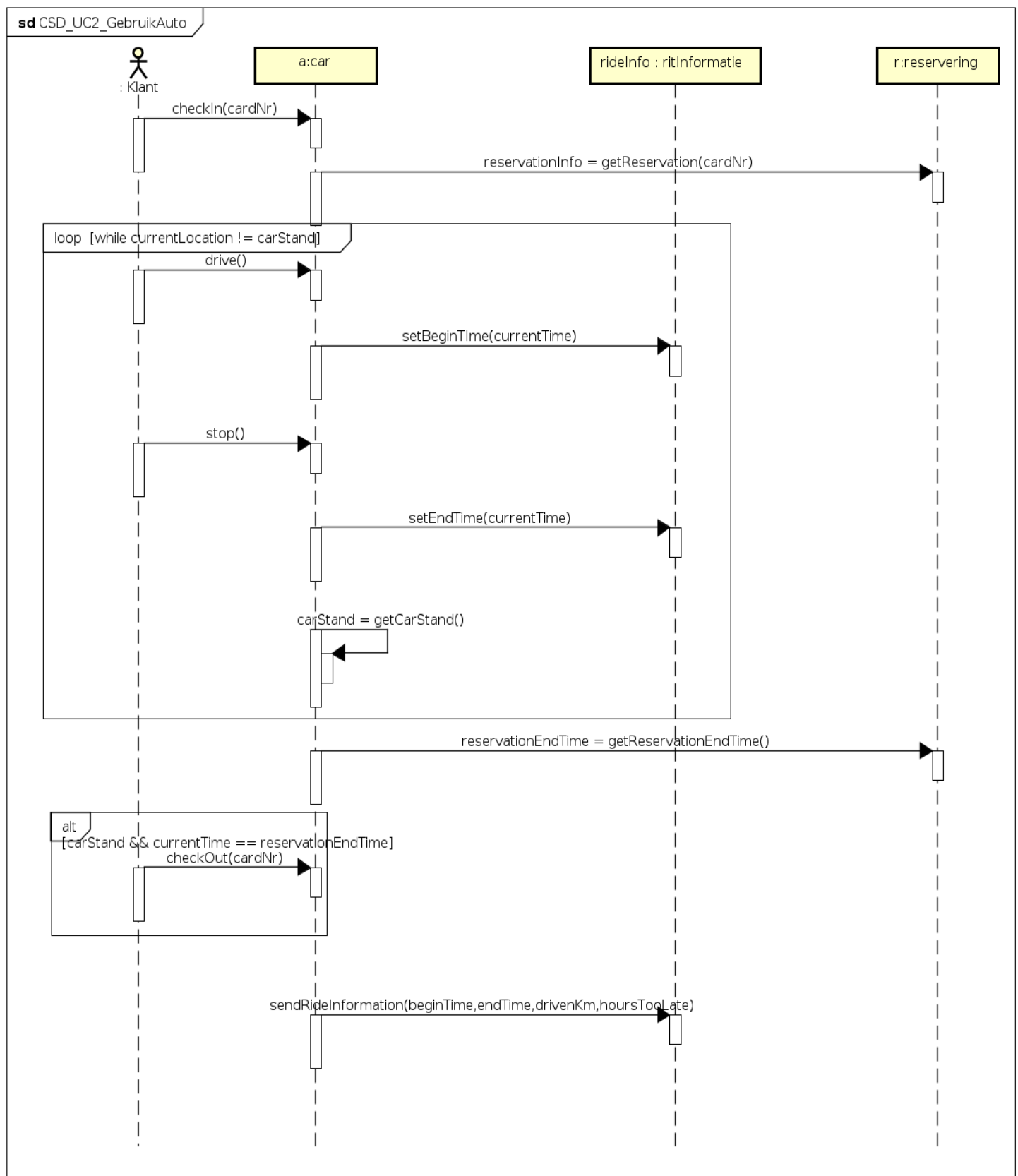
Component Sequence Diagram *Iteratie 1*



CSD_UC2-gebruikAuto v1-0-1

Path to image : CSD_UC2-gebruikAuto_iteratie_1

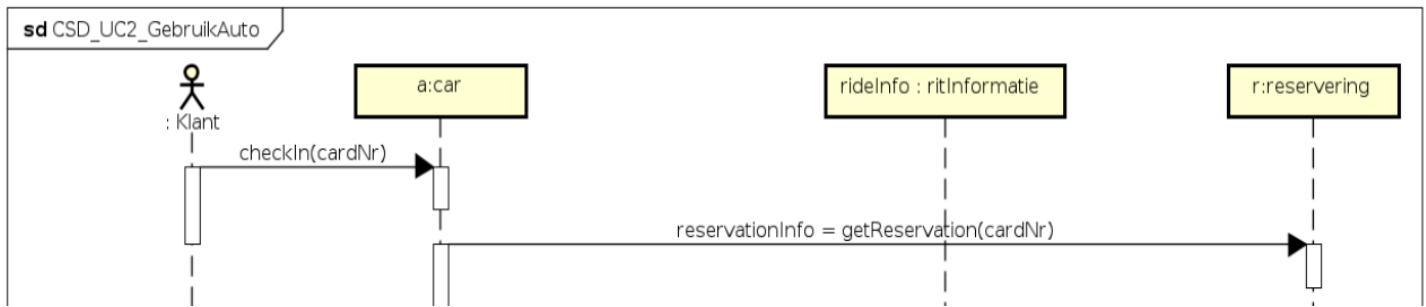
Component Sequence Diagram Iteratie 2



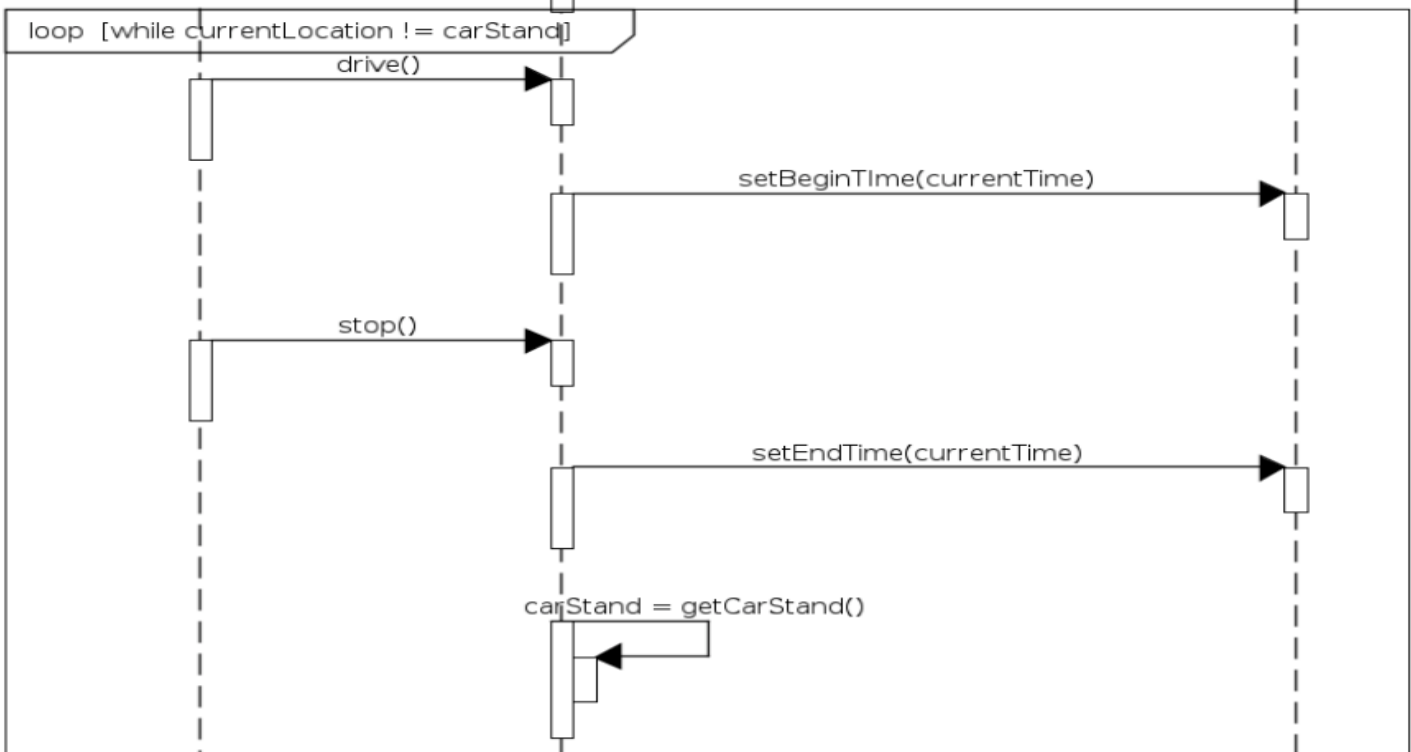
CSD_UC2-gebruikAuto v1-0-2

Path to image : CSD_UC2-gebruikAuto_iteratie_2

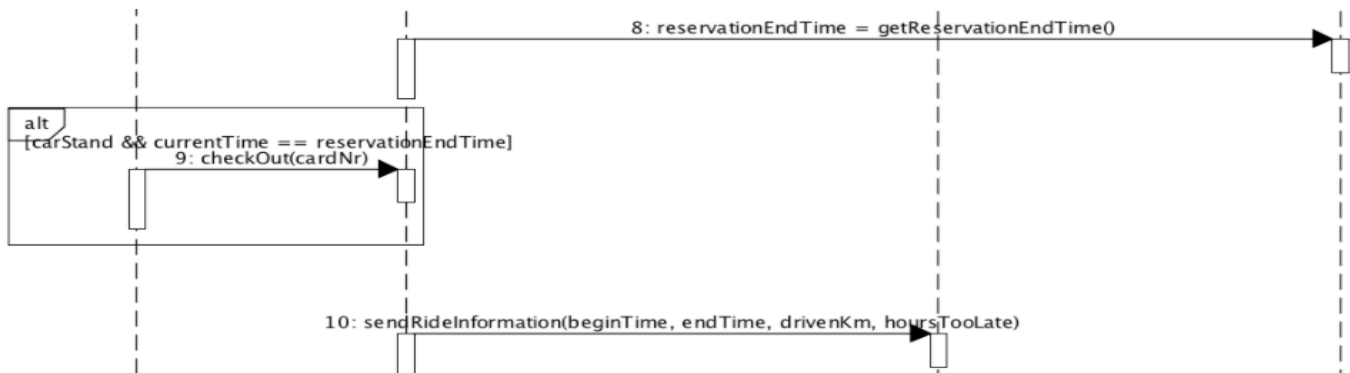
Openen van een auto a.d.h.v. verificatie op de reservering



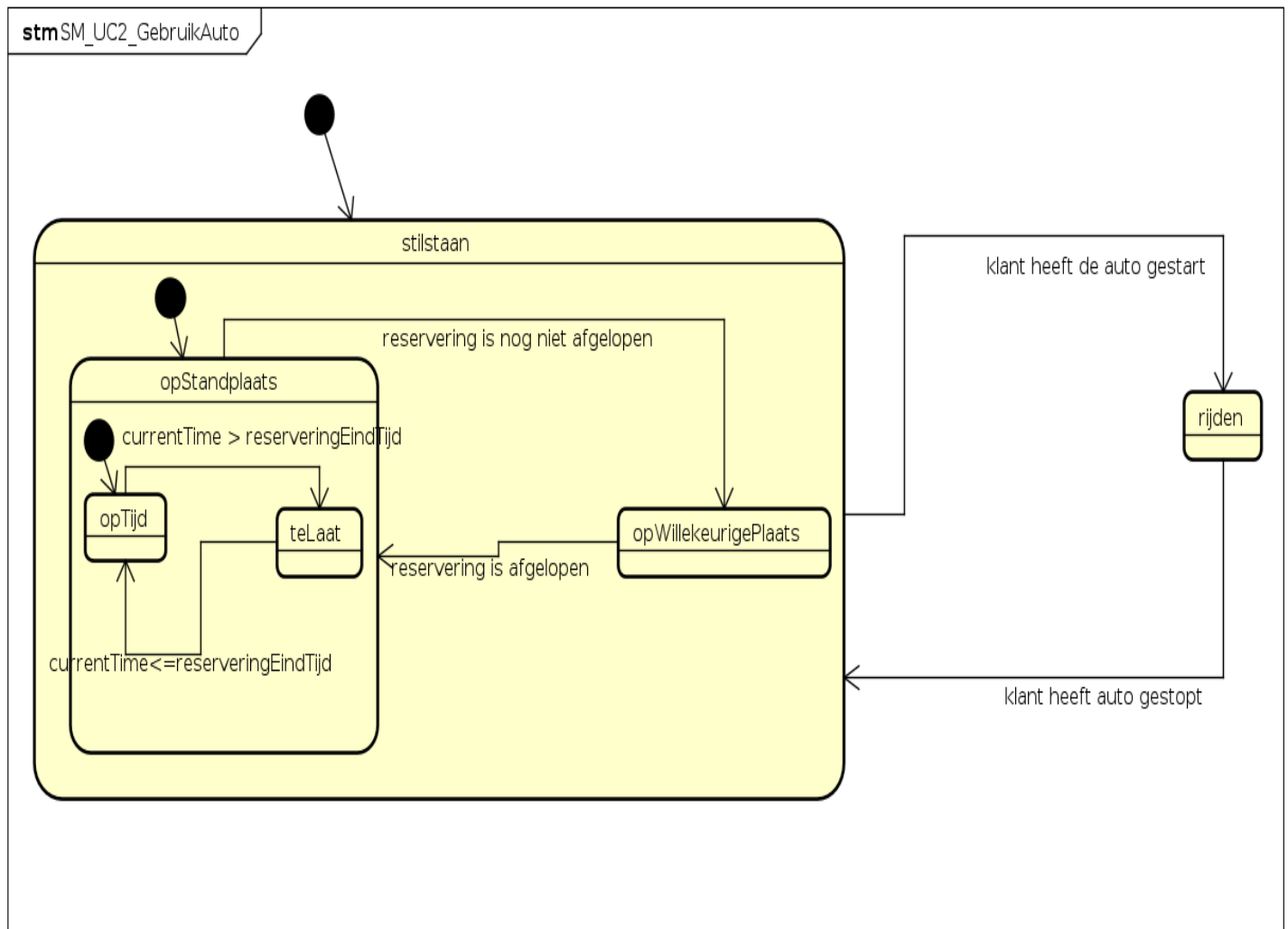
Het maken van 1 of meer ritten tijdens de huurperiode



Het doorvoeren van de gemaakte ritten na de huurperiode.



State Machine Diagram



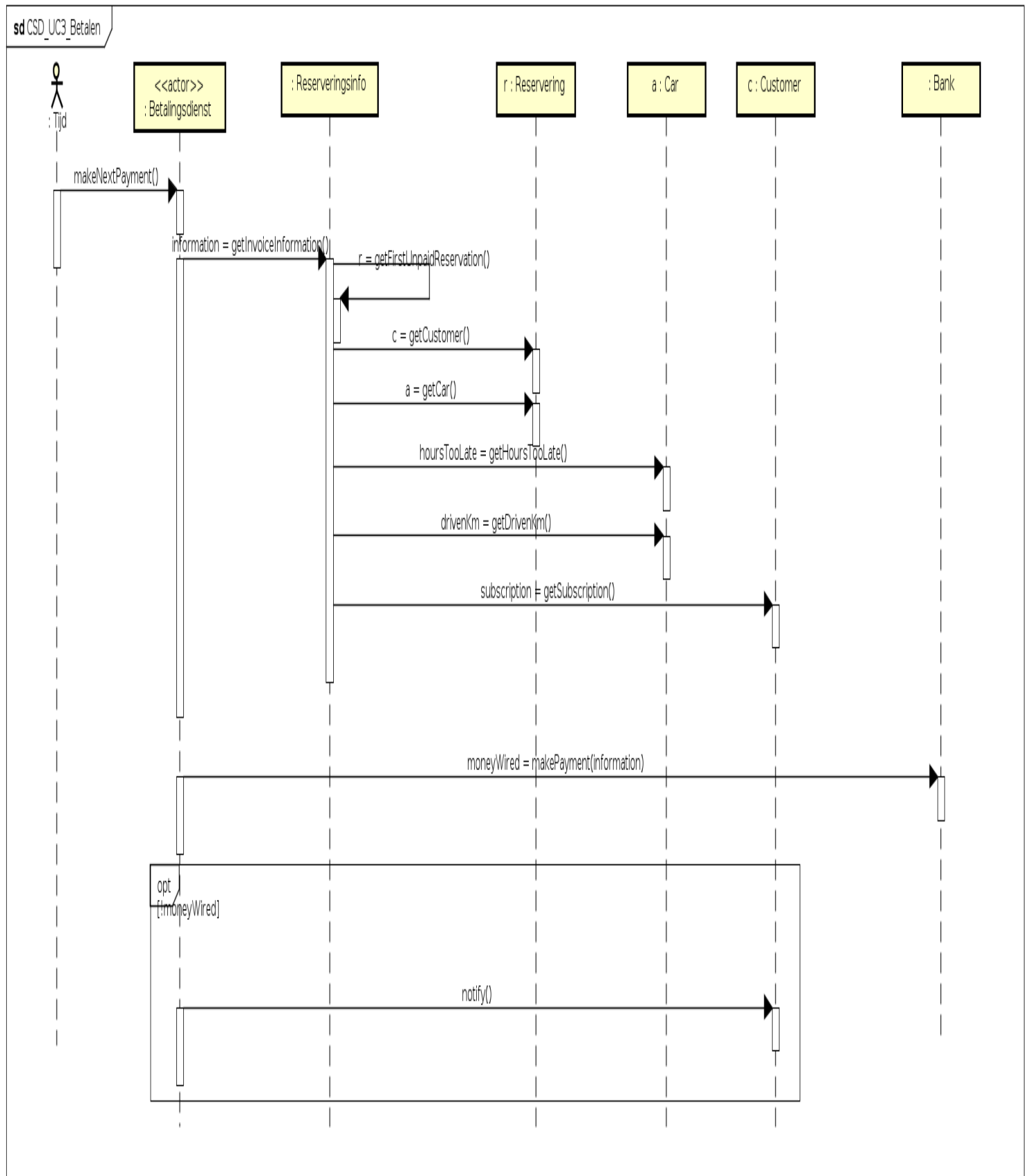
UC 3 Betalingscomponent

Short description

Het betalen van een gehuurde auto gebeurt, zoals vaker benoemd, via een automatische afschrijving.

We hebben tijdens de eerste iteratie een gesteld dat het betalingssyteem actief 'vraagt' aan RedCars welke betalingen vericht moeten worden. Tijdens de tweede iteratie hebben we de actor betalingssyteem veranderd naar van primair naar secundair. Daarbij hebben we tijd als primaire actor toegevoegd (zie [toelichting SRS](#))

Component Sequence Diagram



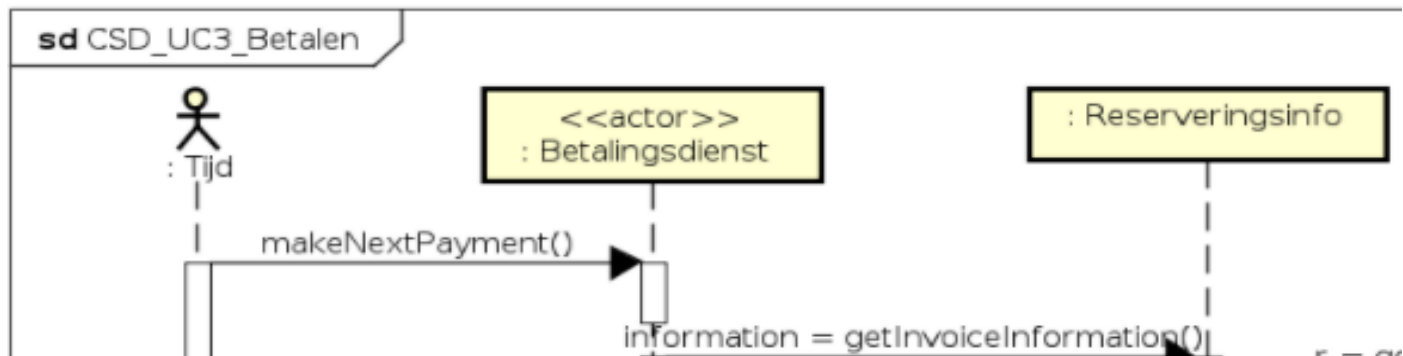
CSD_UC3_Betalen v1-0-2

Path to image : CSD_UC3_Betalen_iteratie_2

Design decisions

**** Van tijd naar betalingsdienst****

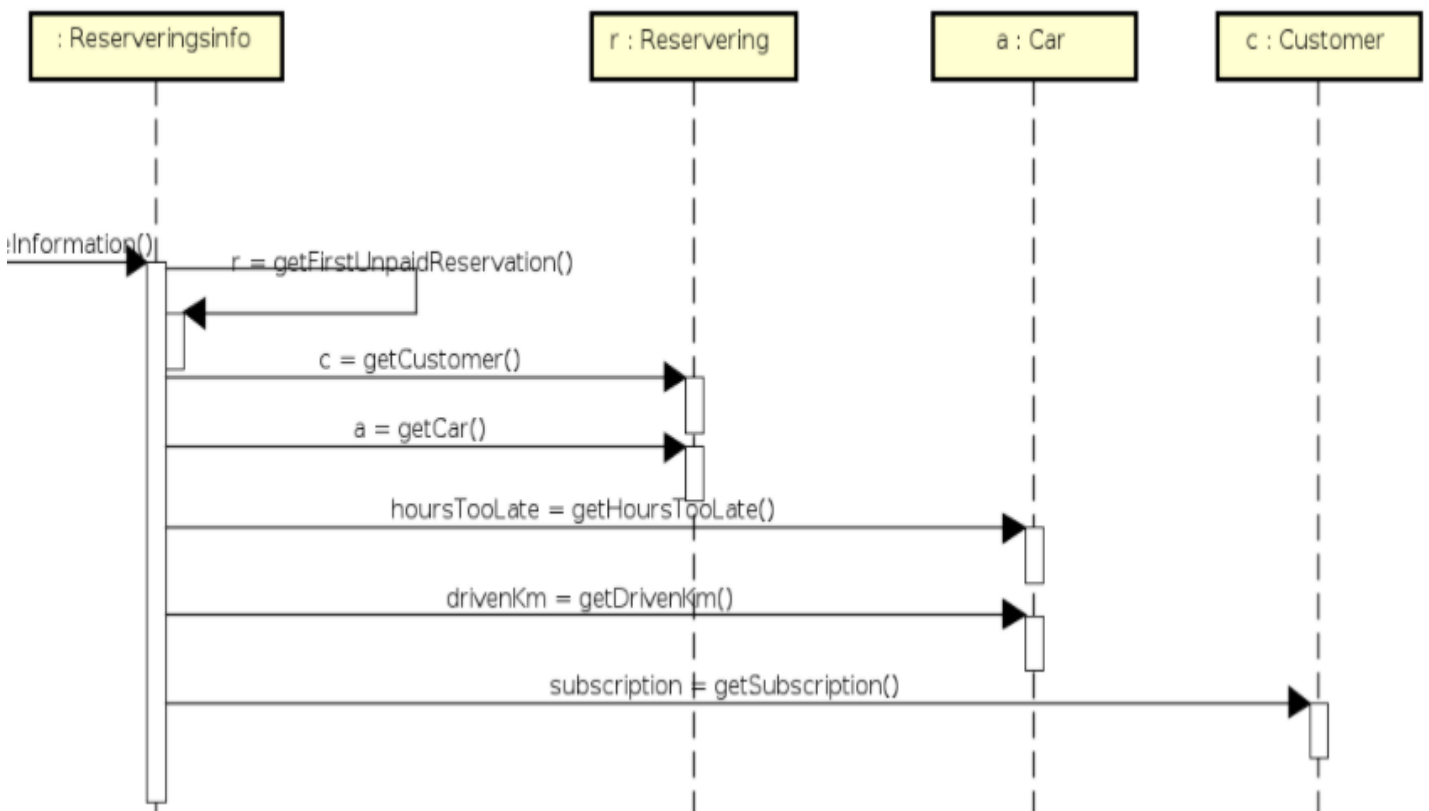
De abstracte actor Tijd triggert de betalingsdienst. De betalingsdienst vraagt aan het Reserveringsinfo component om de volledige betaling te berekenen.



Facade Controller

Wij hebben voor de functionaliteit van het betalingscomponent een probleem opgelost door middel van een Design Pattern. Wij liepen er tegen aan dat het betalingscomponent, na het gebruik van de auto, het totaalbedrag moet weten. Om ervoor te zorgen dat we geen Hogere Coupling en Lagere Cohesie zouden krijgen door het betalingscomponent direct verbinding te laten leggen met de benodigde onderdelen om het totaal bedrag uit te rekenen.

Hierdoor hebben we een 'pure artificial' component bedacht, reserveringsinfo. Dit component is de brug tussen het betalingssysteem en de andere componenten (zie onderstaande afbeelding).

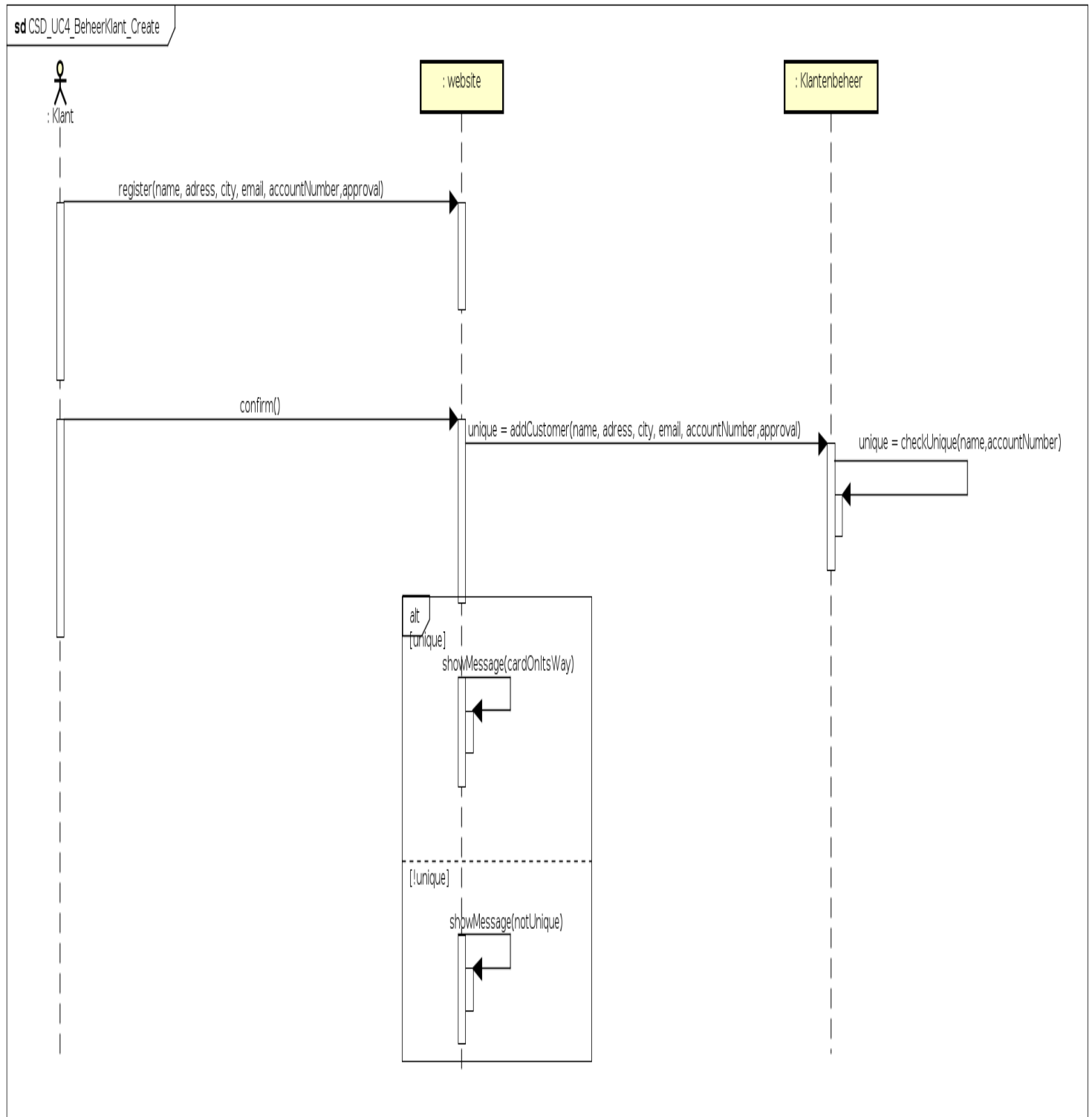


UC 4 Klantcomponent

Short description

Het klant component wordt gebruikt om klanten mee te beheren.

Component Sequence Diagram Create

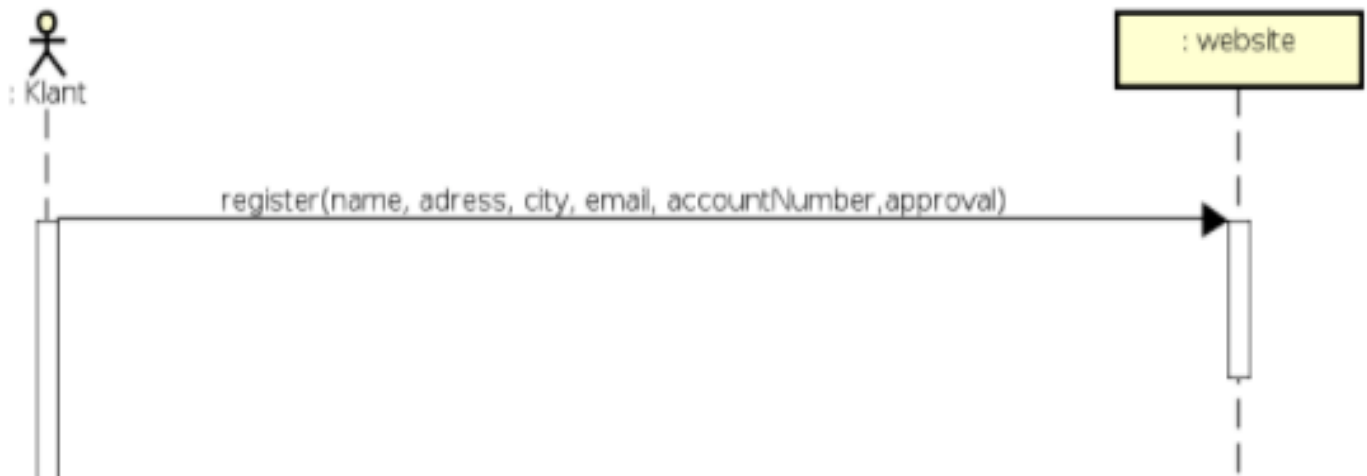


CSD_UC4_BeheerKlant_Create

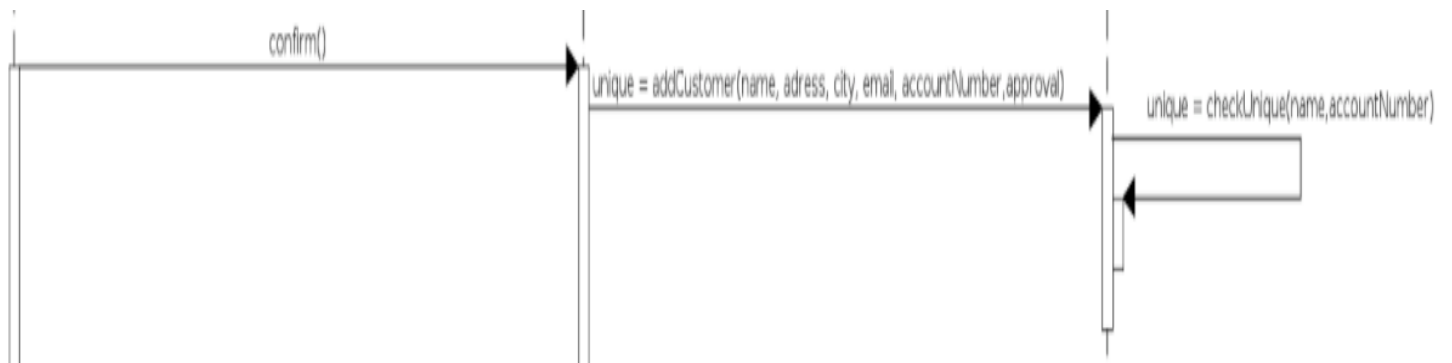
Path to image : CSD_UC4_BeheerKlant_Create

Design decisions

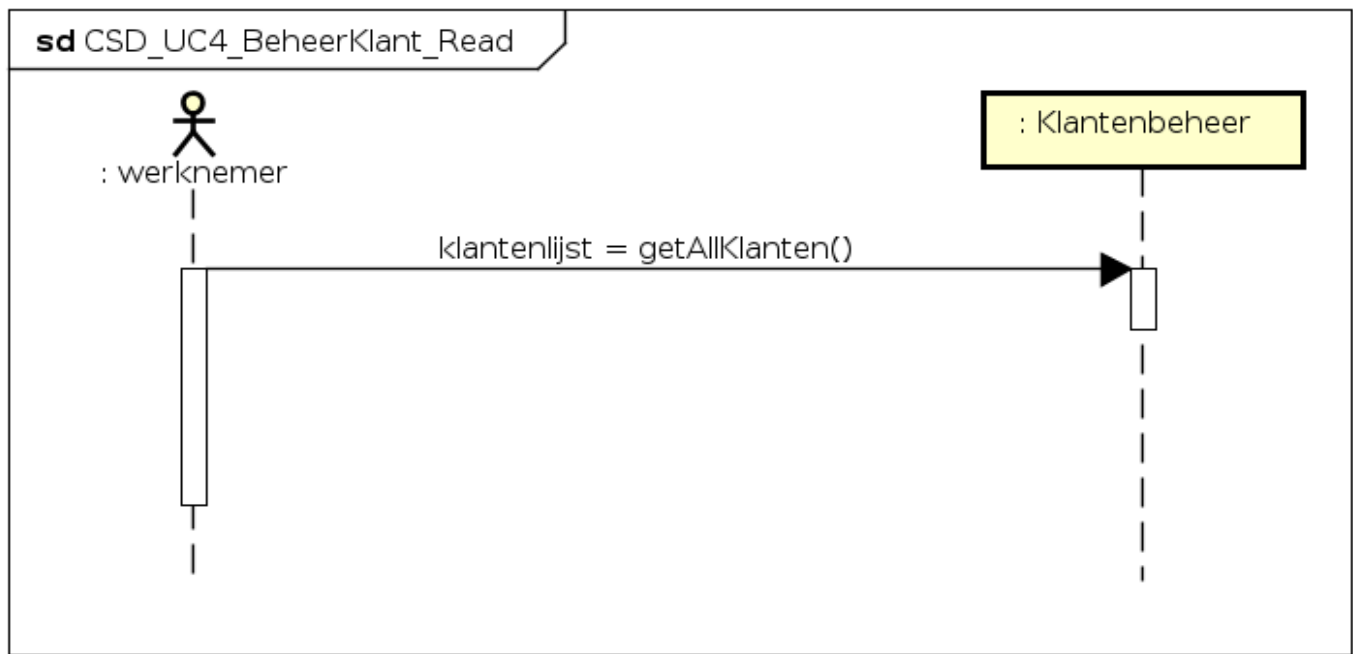
happy flow



Toevoegen nieuwe klant aan systeem en controle op uniek account.

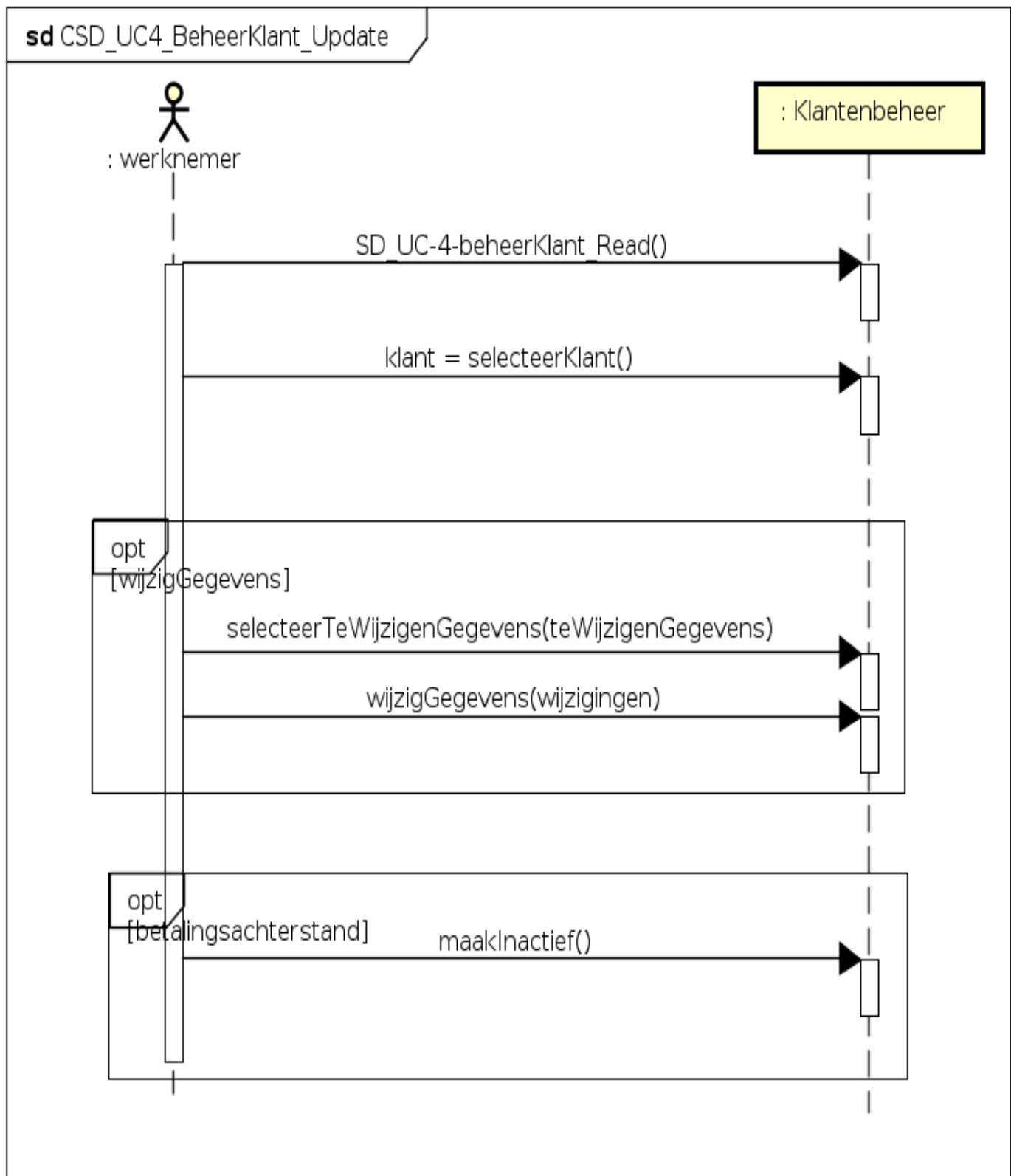


Component Sequence Diagram Read



CSD_UC4_BeheerKlant_Read

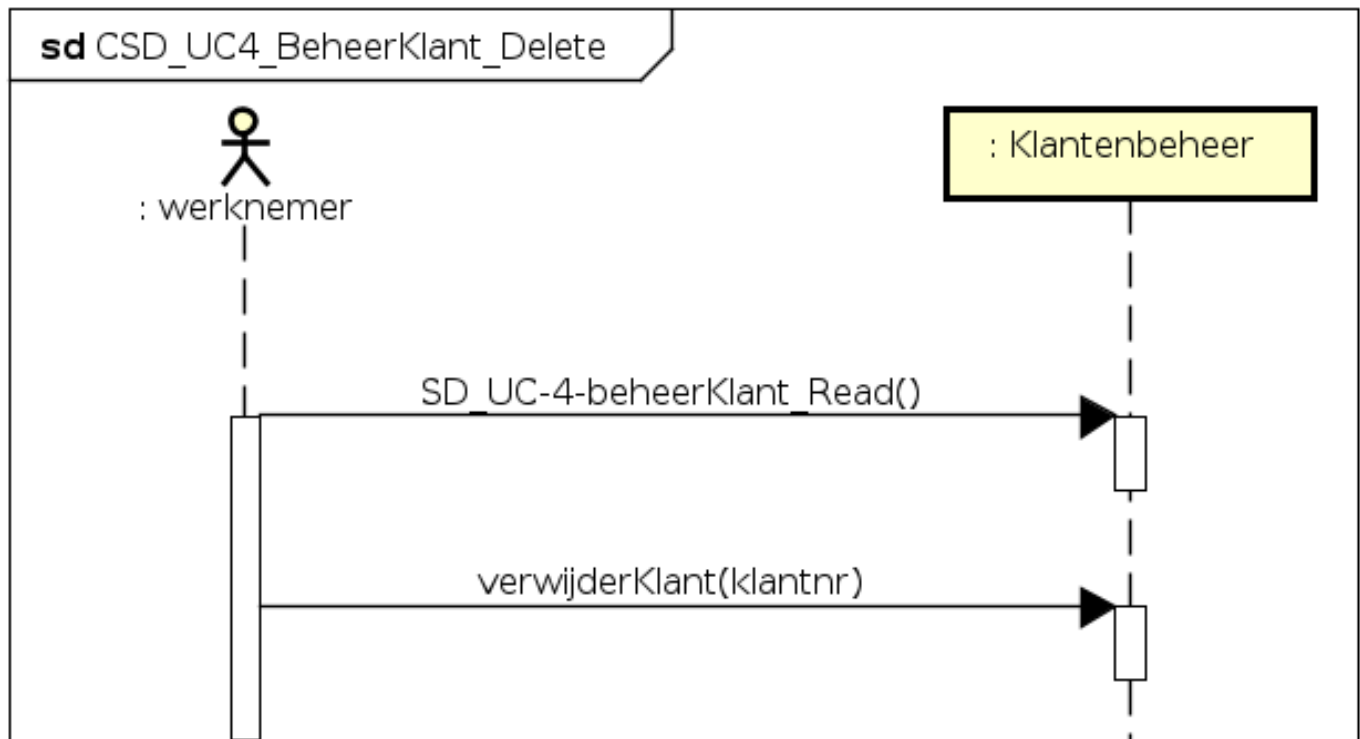
[Path to image : CSD_UC4_BeheerKlant_Read](#)



CSD_UC4_BeheerKlant_Update

Path to image : [CSD_UC4_BeheerKlant_Update](#)

Component Sequence Diagram Delete



CSD_UC4_BeheerKlant_Delete

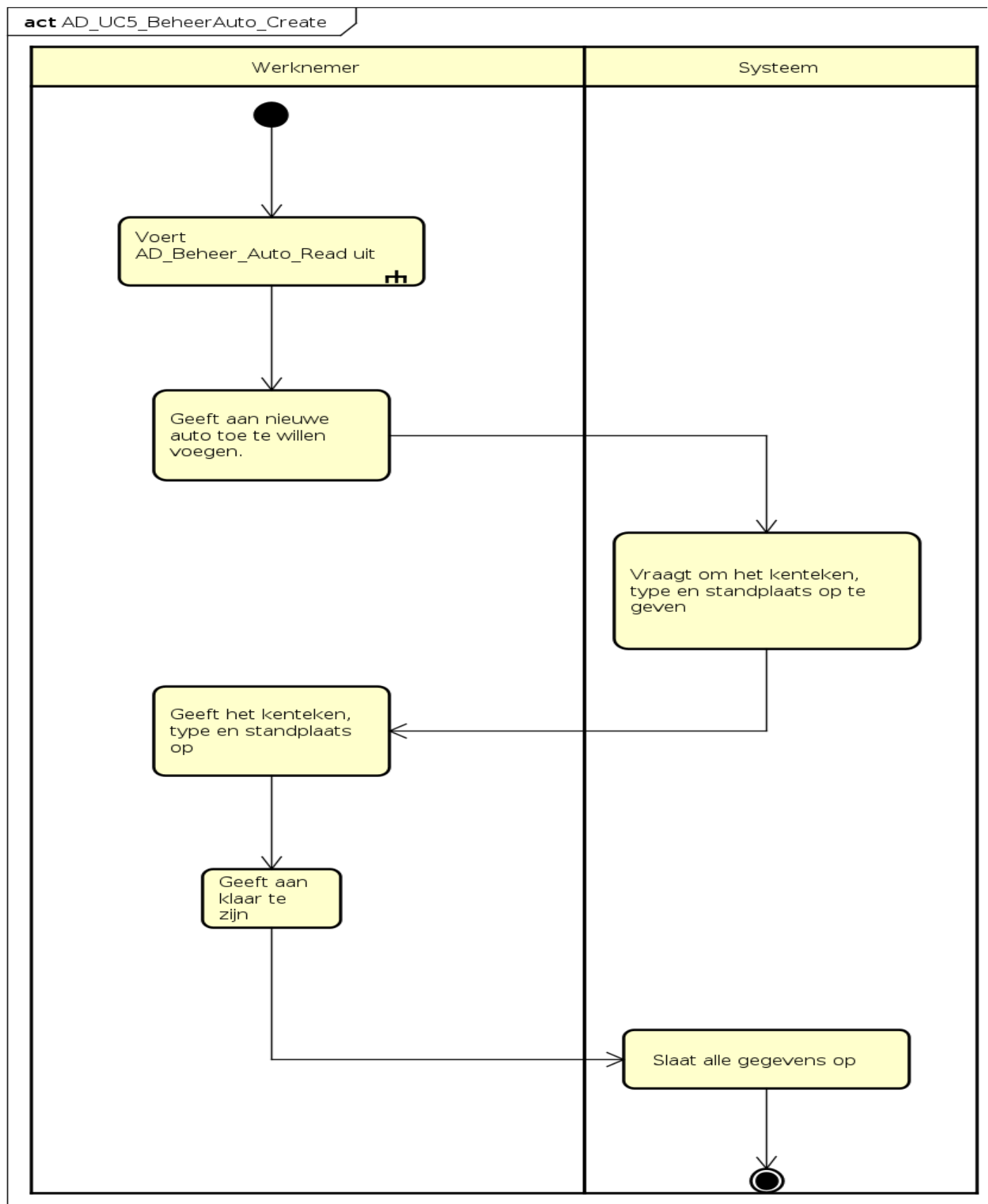
Path to image : [CSD_UC4_BeheerKlant_Delete](#)

UC 5 Autobeheercomponent

Short description

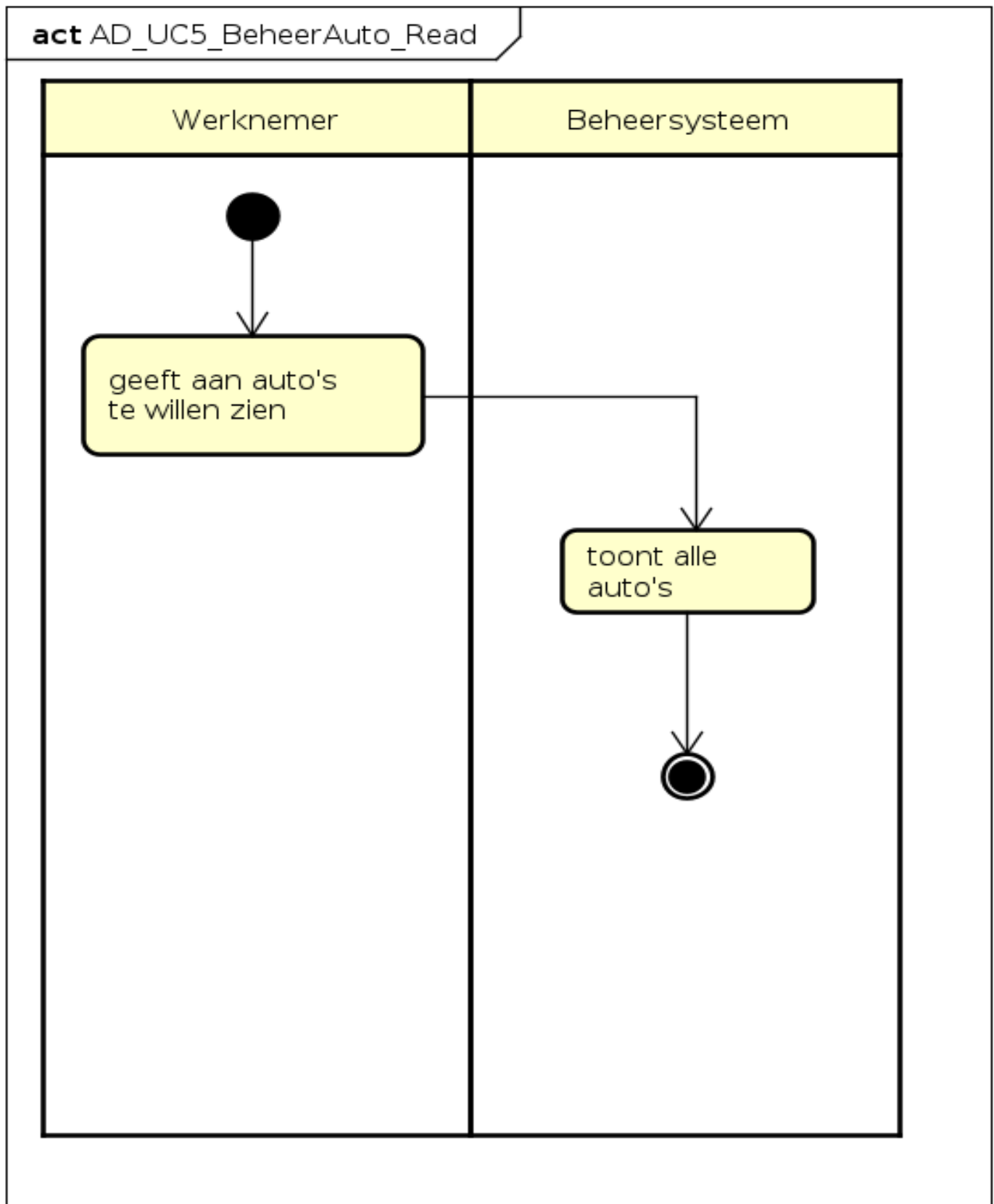
Onderstaande activity diagrams beschrijven de flow van het beheren van een auto.

Activity Diagram Create



AD_UC5_BeheerAuto_Create

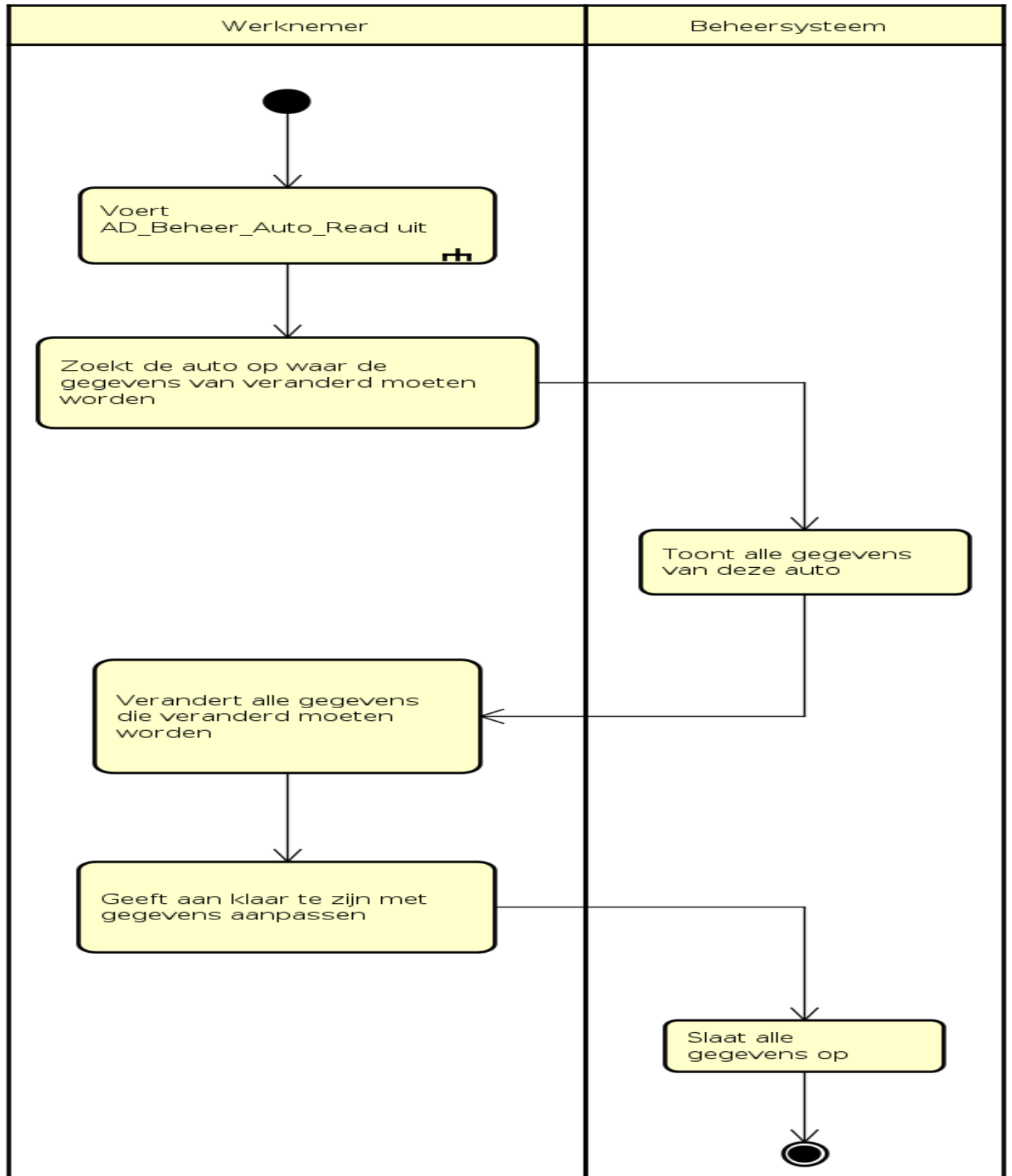
AD_UC5_BeheerAuto_Create



AD_UC5_BeheerAuto_Read

AD_UC5_BeheerAuto_Read

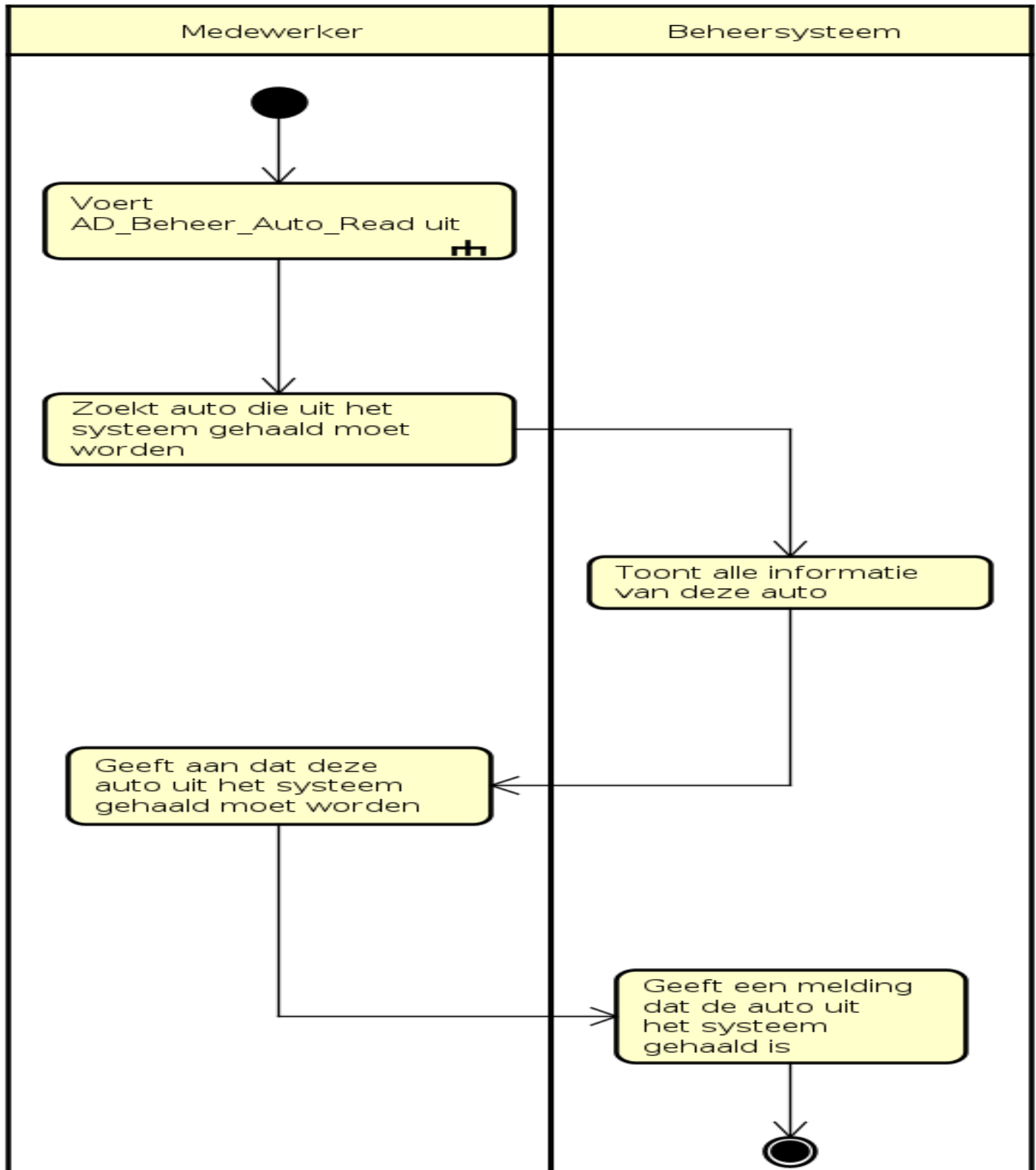
act AD_UC5_BeheerAuto_Update



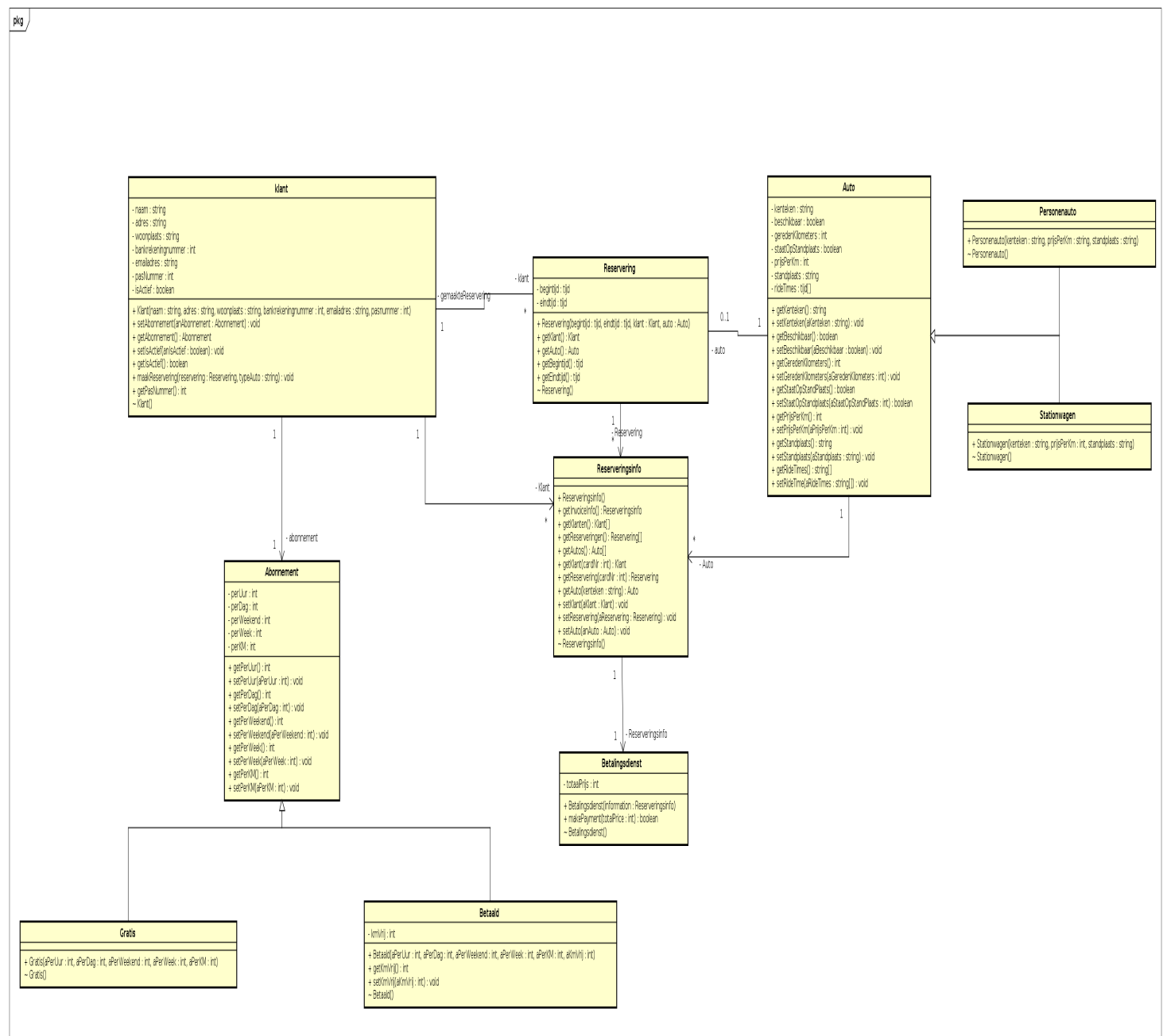
AD_UC5_BeheerAuto_Update

AD_UC5_BeheerAuto_update

act AD_UC5_BeheerAuto_Delete



Design Class Diagram



Design Class Diagram v1-0-0

Path to image : Design Class Diagram_iteratie_2

klant
<ul style="list-style-type: none"> - naam : string - adres : string - woonplaats : string - bankrekeningnummer : int - emailadres : string - pasNummer : int - isActief : boolean
<ul style="list-style-type: none"> + Klant(naam : string, adres : string, woonplaats : string, bankrekeningnummer : int, emailadres : string, pasnummer : int) + setAbonnement(anAbonnement : Abonnement) : void + getAbonnement() : Abonnement + setIsActief(anIsActief : boolean) : void + getIsActief() : boolean + maakReservering(reservering : Reservering, typeAuto : string) : void + getPasNummer() : int ~ Klant()

<i>Abonnement</i>
<ul style="list-style-type: none"> - perUur : int - perDag : int - perWeekend : int - perWeek : int - perKM : int
<ul style="list-style-type: none"> + getPerUur() : int + setPerUur(aPerUur : int) : void + getPerDag() : int + setPerDag(aPerDag : int) : void + getPerWeekend() : int + setPerWeekend(aPerWeekend : int) : void + getPerWeek() : int + setPerWeek(aPerWeek : int) : void + getPerKM() : int + setPerKM(aPerKM : int) : void

Gratis
<ul style="list-style-type: none"> + Gratis(aPerUur : int, aPerDag : int, aPerWeekend : int, aPerWeek : int, aPerKM : int) ~ Gratis()

Betaald

- kmVrij : int

+ Betaald(aPerUur : int, aPerDag : int, aPerWeekend : int, aPerWeek : int, aPerKM : int, aKmVrij : int)
+ getKmVrij() : int
+ setKmVrij(aKmVrij : int) : void
~ Betaald()

Reservering

- begintijd : tijd

- eindtijd : tijd

+ Reservering(begintijd : tijd, eindtijd : tijd, klant : Klant, auto : Auto)
+ getKlant() : Klant
+ getAuto() : Auto
+ getBegintijd() : tijd
+ getEindtijd() : tijd
~ Reservering()

Reserveringsinfo

+ Reserveringsinfo()
+ getInvoiceInfo() : Reserveringsinfo
+ getKlanten() : Klant[]
+ getReserveringen() : Reservering[]
+ getAutos() : Auto[]
+ getKlant(cardNr : int) : Klant
+ getReservering(cardNr : int) : Reservering
+ getAuto(kenteken : string) : Auto
+ setKlant(aKlant : Klant) : void
+ setReservering(aReservering : Reservering) : void
+ setAuto(anAuto : Auto) : void
~ Reserveringsinfo()



Betalingsdienst

- totaalPrijs : int

+ Betalingsdienst(information : Reserveringsinfo)

+ makePayment(totalPrice : int) : boolean

~ Betalingsdienst()

Auto

- kenteken : string

- beschikbaar : boolean

- geredenKilometers : int

- staatOpStandplaats : boolean

- prijsPerKm : int

- standplaats : string

- rideTimes : tijd[]

+ getKenteken() : string

+ setKenteken(aKenteken : string) : void

+ getBeschikbaar() : boolean

+ setBeschikbaar(aBeschikbaar : boolean) : void

+ getGeredenKilometers() : int

+ setGeredenKilometers(aGeredenKilometers : int) : void

+ getStaatOpStandPlaats() : boolean

+ setStaatOpStandplaats(aStaatOpStandPlaats : int) : boolean

+ getPrijsPerKm() : int

+ setPrijsPerKm(aPrijsPerKm : int) : void

+ getStandplaats() : string

+ setStandplaats(aStandplaats : string) : void

+ getRideTimes() : string[]

+ setRideTime(aRideTimes : string[]) : void

Personenauto

+ Personenauto(kenteken : string, prijsPerKm : string, standplaats : string)
~ Personenauto()

Stationwagen

+ Stationwagen(kenteken : string, prijsPerKm : int, standplaats : string)
~ Stationwagen()