POLITECNICO
MILANO 1863

Software engineering 2 Project

Requirement Analysis and Specification Document

PowerEnJoy

Perugini Alex	876359
Re Marco	873564
Scotti Vincenzo	875505

Table of Contents

1 Introduction

1.1 Purpose

This document presents the requirements and specifications of a digital management system for *PowerEnJoy*, an electrical car sharing service. The system allows clients to reserve cars nearby (using GPS position) or in a selected area and use them to move around.
The system will also control cars management, dispatching operators and assigning task to each one according to their availability.
The main purpose of the system is to manage the car sharing service in an efficient way ensuring clients satisfaction and maximum profit.

1.2 Scope

1.2.1 Goals

- [G1] Allows clients to register in the system.
- [G2] Allows users to login.
- [G3] Allows clients to find available cars.
- [G4] Allows clients to reserve only one car at a time.
- [G5] Checks reservation time limit.
- [G6] Allows client to get into the reserved car.
- [G7] Charges the client according to the duration of the ride.
- [G8] Allows the clients to know the actual charge.
- [G9] Identifies correctly the duration of a ride.
- [G10] Rewards or punishes clients according to their behavior.
- [G11] Notifies operators when their intervention is needed.
- [G12] Allows client to use money saving option.

1.2.2 Domain assumptions

- Operators are provided with special login elements from the company.
- Operators are provided with a smartphone from the company.
- Client closes the doors of the car after exiting from it.
- Cars are provided with a satellite navigation inside.
- Cars have a unique identification code which is shown on the windscreen.
- Cars are provided with a GPS system.
- System always detects correctly if the engine of a car is switched on or off.
- System always detects the number of people in the car correctly.
- System always detects correctly if the doors of the car are opened or closed.
- System always detects correctly a car position through GPS system.
- System always detects correctly client’s position if GPS is turned on.
- System has the set of safe area positions in his data base.
- System always detects the battery level of a car correctly.
- System always detects correctly if a car is charging.
- System always detects correctly which plugs of a charging station are in use.
- System always detects correctly if a car is damaged or has a breakdown.
- Car GPS cannot be switched off.

- Operators GPS is switched on during the whole work time.
- Smartphones provided to the operators will respond to the hardware constraints.

1.3 Glossary

1.3.1 Glossary

Charging Area/Station	Subset of safe area where is possible to plug the car to a power grid to charge its battery
Client	User of PowerEnJoy registered to the system with normal log in elements
Driver	User sitting in the car that has the control of the vehicle, can be both a client or an operator
Driving Licence	Category B licence needed to drive the cars provided by PowerEnJoy
Money Saving Option	Functionality provided through the navigation system that enlights the nearest charging area to the client destination with free plugs
Operator	Employee of the company that provides the PowerEnJoy service, he is in charge of managing the cars according to the instructions he receives from the system
Passenger	Person inside the car that is not on the driver seat
Payment Method	The payment methods are all the methods accepted by the external payment system that the system will interface with
Power Grid	Grid located at each power station, it has plugs to charge the PowerEnJoy cars
Punishment	Fee on the charge for the ride
Ride	Entire operation of driving the car from the moment the engine is ignited to the moment the car is left parked in a safe area
Reward	Discount on the charge for the ride
Safe Area	Area defined by a set of GPS positions in which is possible to park the PowerEnJoy cars
Special login elements	Email and password gave to operators to authenticate as employes during the worktime

1.3.2 Acronyms

GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
RASD	Requirement Analysis and Specification Document

1.3.3 Abbreviations

App	Application, in this case refers to the software installed on smartphones

1.4 Reference Documents

1. Project description: Assignments AA 2016-2017.pdf
2. Document standards: IEEE standard on requirement engineering.pdf
3. Alloy model: PoweEnJoy.als
4. Example documents:
    - RASD sample from Oct. 20 lecture.pdf
    - 2012_project_RASD_example_SWIMv2.pdf

1.5 Overview

The document is organized as follows:

- Section 1: Introduction, provides a general description of the intended product with particular stress on the purposes, the goals and the context the system will be inserted in.
- Section 2: Overall Description, gives a detailed description of the software functionalities, focusing on the ways it will relate to clients and its requirements.
- Section 3: Specific Requirements, contains the results generated by the analysis of requirements with the intended ways of use and development patterns.
- Section 4: Alloy, contains the model specification for the system and the results obtained by the software alloy used to improve requirements and model analysis.
- Section 5: Used tools, contains the list of the tools used to realize the RASD and the scope they were used for.
- Section 6: Hours of work, report of work time for each member.

2 Overall Description

2.1 Product Perspective

2.1.1 System Interfaces

The system has to interface with an external Payment System, that allows clients to pay for the service, and the Motorization system, that allows PowerEnJoy to verify the driving licenses of the clients.The Payment System will manage all the payment issues and notify the system according to the results while the motorization system will be only used to validate the licenses of clients.

2.1.2 User Interfaces

Client can interface the system through a smartphone application in which he/she can take advantage of all provided functionalities. In particular, at first only register/login screen is shown, with the optional mark to keep log in saved on the bottom of the same screen. In the registration screen clients will be asked to fill in a form with their personal informations (name, surname, address, license id, payment method and email address), once registration is correctly completed they will get back to the registration/login screen, otherwise client will get an error message and will remain in the registration page. Once registration is confirmed, client receives a password that can be used to log in the system. After login has been executed, clients can see the map with available cars under which will appear two buttons: one to search cars near to their position (in this case, if not yet enabled, the client will be asked to enable GPS), and the other one to search the cars near a specified position they will be asked to provide. In both cases the client will get to a new screen with the list of available cars, listed by distance from the given position.
After selecting the car, the client can reserve it through a button on the popup menu that appears. Once the car is reserved a timeout of one hour starts so that the client can know when his/her reservation expires. Under the timeout will be displayed one button that allows the client to notify the system he/she is near the reserved car.
The application has also a pop-up menu, available in every screen, where the client can manage the account: logout button, settings, payment history and personal informations management.
In the payments area the client can see all the reservations with the related payment and optionally the reward received for that ride.
In the personal informations management the client can modify his/her personal informations such as license code or payment method in order to keep the system up to date. The eventual changes will take place in the same way of the registration both for the user that will fill a new form and for the system that will validate again the new informations.
The client interfaces the system also through the screen of the satellite navigator put in the car. During the ride the screen shows to the client the actual charge and, if requested by the client, shows also the power station in which leave the car in order to get a discount.
The operators of PowerEnJoy can access the reserved area of the application in which they can see a list of cars that need their technical intervention. When the operator selects a car the application shows the position on the map and the type of intervention needed (re-

charge, move from inappropriate parking and so on). In any way when a car needs an immediate intervention they will receive a notification through the app.

2.1.3 Hardware Interfaces

The hardware interfaces for the intended product that will interact with clients and operators are going to be the smartphones (operators will be provided by the company for the ones to use during work) and the screen of the satellite navigation inside every car.

2.1.4 Software Interfaces

The system has to interface also with a Data Base Management System which manages all user's data and other internal data (e.g. cars information, location of the stations, etc..).

2.2 Product Functions

2.2.1 Requirements

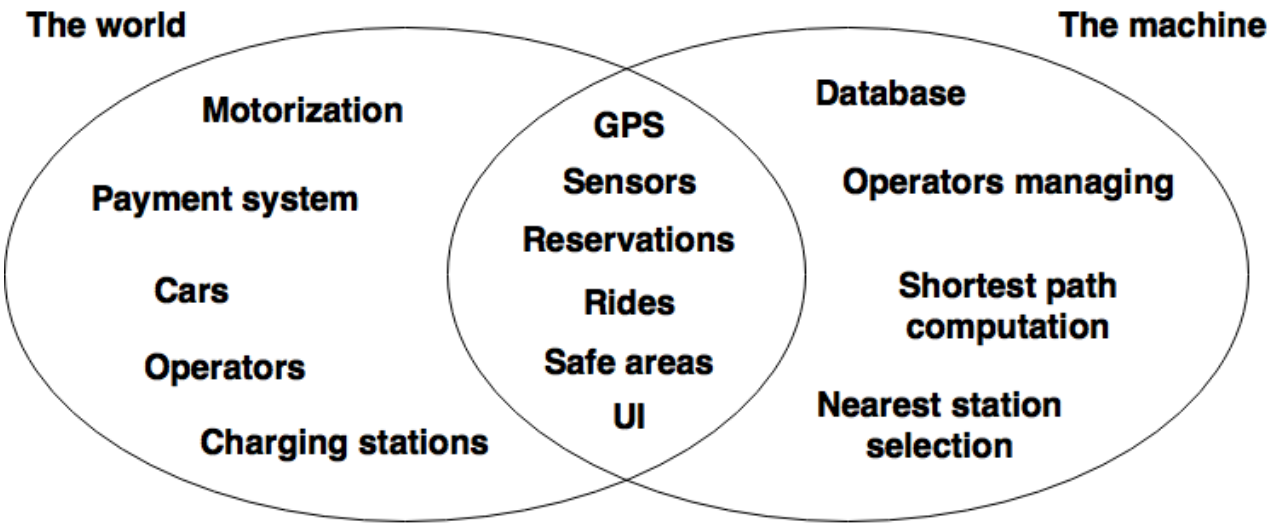
- [G1] Allows clients to register in the system
  - The system must accept the registration only if all the required fields are filled
  - The system must save client's data in the database
  - The system must send a request to the motorization to check the validity of the driving license
  - The system must send a request to the payment system to check the validity of the payment method
  - The system must show an error message if the inserted data are wrong or the fields are filled incorrectly
- [G2] Allows users to login
  - The system must show an error message if the credentials are wrong
  - The system must permit client to access to clients' area if the credentials are correct
- [G3] Allows clients to find available cars
  - The system must show the map with the available cars nearby if the GPS is turned on
  - The system must show the available cars near the position inserted by the client
  - The system must show an error message if the inserted position is not valid
  - The system must refresh the map if a car becomes unavailable or if a car becomes available
- [G4] Allows clients to reserve only one car at a time
  - The system must show a confirmation message if the reservation has been completed successfully
  - The system must show an error message if the client has already reserved another car
  - The system must save the time at which the reservation has been carried out
- [G5] Checks reservation time limit
  - The system must delete the reservation if the car isn't picked up within one hour

    - The system must mark a car as available when its reservation expires
- [G6] Allows clients to get into the reserved car
    - The system must let the client choose if he/she wants to unlock the car according to his/her position or inserting the code on the windscreen
    - The system must unlock the car if the client has chosen the position option and the detected position is near the position of the car
    - The system must unlock the car if the client has chosen the code option and the inserted code is correct
    - The system must show an error message if the client has chosen the code option and the inserted code isn't correct
    - The system must show an error message if the client has chosen the position option and the detected position isn't near the position of the car
- [G7] Charges the client according to the duration of the ride
    - The system must start charging the client when the engine ignites
    - The system starts charging the client anyway if he doesn't ignite the engine within one minute
    - The system must increase the charge every minute
- [G8] Allows the clients to know the actual charge
    - The system must show the charge for the ride on the screen inside the car
    - The system must update the actual charge during the ride as stated in the point 3 of [G7]
- [G9] Identifies correctly the duration of a ride
    - The system must be able to recognise whether a car is in a safe area or not
    - The system must display an error message on the screen of the satellite navigation if the car is switched off outside a safe area
    - The system must advice with a beeper that the door of the car has been opened in an unsafe area
    - System must lock the car if the ride is finished
    - System must stop the charging when the ride is finished and the car is in a safe area
    - System must consider a ride finished if and only if the car is left empty inside a safe area
- [G10] Rewards or punishes clients according to their behavior
    - System must charge a fee of 1€ to the client in case the car he reserved is not picked up within an hour from the reservation
    - System must apply a 10% discount on the total cost of the ride if the client takes at least other two passengers
    - System must apply a 20% discount on the total cost of the ride if the car is left with no more than 50% of the battery empty
    - System must apply a 30% discount on the total cost of the ride if the car is left at a power station and the client takes care of plugging the car into the power grid
    - System must apply a 30% additional fee on the total cost of the ride if the car is left at more than 3 KM from the nearest power grid station or with more than 80% of the battery empty
    - System must inform the clients of their rewards/punishments through the app
- [G11] Notifies operators when their intervention is needed

- The system must alert the nearest operator when a car is left for more than one minute empty and switched off out of a safe area
- The system must notify the nearest operator when a car is damaged or has a breakdown
- System must lock the car while waiting the operator
- System must let only the notified operator enter inside the car
- The system must notify the nearest operator when a car is left for more than one minute with more than 80% of the battery empty informing him/her if there is a reachable charging station or if it's necessary to recharge on-site
- The system must notify the nearest operator 3 minutes after a car is left in a charging station without the plug
- System must add the notification in a queue if there are no available operators

- [G12] Allows client to use money saving option
  - System must display 'Money Saving' button on the screen of the car
  - System must require the destination to the client when the money saving option has been selected
  - System must calculate and sort charging areas near to the destination
  - System allows client to select the desired charging station
  - System provide assistance to navigation to reach the destination
  - System displays an error message if the provided address for the destination is wrong

2.2.2 The World and the Machine



This graph represents the distinction among things that are in the world, in the machine to be created and shared between these two.
The motorization and the payment system are external systems that interface PowerEnJoy system so are located in the world. Physical object such as cars and charging stations are

also located in the world. The operators are employees of PowerEnJoy so they are in the world.
Database, operators managing, shortest path computation and nearest station selection are all data or operations that are exclusively located in the machine which is in charge of use and modify them.
In the shared phenomena there are GPS and sensors which collect informations from the world providing usable data and informations to the machine. The UI is also a shared phenomena because it allows the machine to communicate with its users. Lastly the concepts of safe area, reservation and ride, these three are things that are in the world and can be observed, used or created by the machine.

2.3 User characteristics

System will distinguish between two kinds of users: clients and operators

2.3.1 Clients

Generic clients of the system

Function	Reserve and drive the shared cars
Type of Device	Smartphone
Required characteristics	PowerEnJoy application installed on the smartphone, Class B driving licence, email address, credit card, ability to read maps
Nature of use of the system	Necessity to reach some place

2.3.2 Operators

Persons delegated by the owners of the system

Function	Manage shared cars on request
Type of device	Smartphone
Required characteristics	Same as the driver, but they also need to be registered employees
Nature of use of the system	Receive work instructions

2.4 Constraints

For what concerns the hardware constraints the mobile application requires at least a 3G internet connection or a Wi-Fi connection, space for app installation on the smartphone, GPS connection is optional but without it the client will not be able to use all the system functionalities (car search through GPS position).

3 Specific Requirements

3.1 Functional Requirements

3.1.1 Scenarios

3.1.1.1

Name	Registration in the system
Goal	[G1] Allows client to register in the system
Assumptions	1. Client is not registered
John needs a way to visit his friend Anna and decides to use the PowerEnJoy service. He never used this service before so, after downloading the app, he has to register in the system by tapping on the 'Register' button. In order to complete the registration John has to compile a form with his data: name, surname, email, password and driving license code and then press the 'Submit' button. Now the registration is complete and John can access the system and reserve a car to visit his friend.	

3.1.1.2

Name	Login the system and try to reserve more than one car
Goal	[G2] Allows users to login [G3] Allows clients to find available cars [G4] Allows clients to reserve only one car at a time
Assumptions	1. Client is registered in the system 2. Client is not logged in 3. Client's GPS is turned on
Mario has to go to work but his car is being repaired so he decides to login the PowerEnJoy app inserting his email address and password and pressing the 'Login' button. Now he can see his position on the map and all the nearby cars, he chooses one of them and a pop-up window with 'Yes' and 'No' options appears, asking if he wants to reserve that car. Mario tap the 'Yes' button and receive a confirmation message that the reservation has been completed. Some minutes after Mario reopens the app and notices on the map that there is a new available car nearer to his position than the one which he has reserved, so he tries, with the procedure described above, to reserve this new car but the system prevent him to do the reservation with a pop-up message saying: "You cannot reserve more than one car at the same time".	

3.1.1.3

Name	Expiration of the reservation

Goal	[G5] Checks reservation time limit [G6] Allows client to get into the reserved car
Assumptions	1. Client is registered 2. Client is logged in 3. Client's GPS is turned on
Jennifer wants to use PowerEnJoy to go to a party so she chooses a nearby car on the map and tap the 'Reserve' button to reserve it, at the top of the screen a timeout of one hour appears. Unfortunately she is late with all the things she has to do and more than one hour passes from the moment of the reservation, so when she reopen the app can notice that the timeout expired and a pop-up message appears saying that the reservation is lost. Once pressed the 'Ok' button the message disappears and Jennifer sees that the car is still available, she reserves it and goes to pick it up. Once arrived she opens the app and presses the 'Unlock' button, the system verifies that her position is near the car and unlocks it. Jennifer get into the car and goes to the party.	

3.1.1.4

Name	Unlocking of the car
Goal	[G3] Allows clients to find available cars. [G4] Allows clients to reserve only one car at a time. [G6] Allows client to get into the reserved car.
Assumptions	1. Client is registered 2. Client is logged in
Brian wants to reserve an electric PowerEnJoy car in a specific place, so he fills the bar with an address and presses the 'Ok' button. The map is now centered on the address inserted and Brian chooses one of the available cars to reserve it. Once arrived to the car Brian opens the app and presses the 'Unlock' button. The system asks the client to insert the code printed on the windscreen of the car. So Brian inserts it and submits. The system verifies it is correct and then, if so, unlocks the car and notifies it in the app. Now Brian can enter and go where he wants.	

3.1.1.5

Name	Use of the car
Goal	[G7] Charges the client according to the duration of the ride. [G8] Allows the clients to know the actual charge. [G9] Identifies correctly the duration of a ride. [G10] Rewards or punishes clients according to their behavior.
Assumptions	1. Client is in the reserved car
Cristina wants to meet her friend Joseph in the center of the city and just got in the PowerEnJoy car to reach him. When she turns on the engine the also the screen inside the car turns on and the charge starts. During the ride she can see the charge growing up on the screen. Once reached the center of the city Cristina parks the car in a safe area	

and goes out of it. Cristina accesses the application and goes through the menu to the payments, she can see the amount paid for her ride and notice that a discount of 20% is applied because she left the car with less than 50% battery empty.

3.1.1.6

Name	Discount due to passengers
Goal	[G9] Identifies correctly the duration of a ride. [G10] Rewards or punishes clients according to their behavior.
Assumptions	1. Client is in the reserved car with two other passengers
Bob decided to go to the cinema with his friends Josh and Marta, so he reserved a PowerEnJoy car and now he is looking for a park near the cinema. Once parked the car the passengers and the driver go out to see the movie, while walking into the cinema Bob notices on his PowerEnJoy app that he received a 10% discount for taking two passengers in the car with him.	

3.1.1.7

Name	Only one discount can be applied
Goal	[G9] Identifies correctly the duration of a ride. [G10] Rewards or punishes clients according to their behavior.
Assumptions	1. Client is in the reserved car with two other passengers
Marta is going to the university with two classmates using a PowerEnJoy car. Near the university there is a special parking area in which PowerEnJoy cars can be recharged, so she decides to park in that area and plugs the car into the power grid. Then she looks on the app and notices that has achieved two awards for her virtuous behaviour and the higher discount has been applied to the last ride, in this case 30% for plugging the car into the power grid.	

3.1.1.8

Name	Additional fee
Goal	[G9] Identifies correctly the duration of a ride. [G10] Rewards or punishes clients according to their behavior.
Assumptions	1. Client is in the reserved car
Richard is going to a work meeting with a PowerEnJoy car and is late. He is in a hurry so leaves the car in a safe area without caring about the nearest power grid station (that was more 5 Km distant). After the meeting he opens the payment section the PowerEnJoy application where he finds out that an additional charge of 30% has been applied on his last ride, with a message saying that he left the car with less than 20% battery or more than 3 Km far from the nearest power grid station.	

3.1.1.9

Name	Money saving
Goal	[G9] Identifies correctly the duration of a ride. [G10] Rewards or punishes clients according to their behavior. [G12] Allows client to use money saving option
Assumptions	1. Client is in the reserved car
Mary is coming back home with a PowerEnJoy car and she wants to get a discount so presses the ‘Save Money’ button on the screen of the car, now she has to insert her destination address and then click the ‘Ok’ button. A map appears on the screen to show the position of a plug station which is distant 73 meters from the inserted address. Mary parks the car in the suggested station, plugs the car and gets a 30% discount on the ride.	

3.1.1.10

Name	Wrong area parking
Goal	[G9] Identifies correctly the duration of a ride.
Assumptions	1. Client is in the reserved car 2. Client notices the warning on the satellite navigation screen 3. Client does not get out of the car
Andrew is driving to the drug store using a car from PowerEnJoy. Suddenly he finds a free parking, but it isn’t registered in the safe areas. Ignoring this information he parks the car, switches off the engine and opens the door. Straight after opening the door, beeper turns on and he finds out the notification on the screen of the car that informs him the car is not parked in a safe area and that the charging is still going on. Immediately he switches on the car, finds a safe area through the app and drives the car to the correct parking. This time when he gets out the system registers the end for his ride.	

3.1.1.11

Name	Operator relocates car correctly
Goal	[G9] Identifies correctly the duration of a ride. [G10] Rewards or punishes clients according to their behavior. [G11] Notifies operators when their intervention is needed
Assumptions	1. Client does not leave the car in the safe area 2. Client ignores the notification for the wrong parking and leaves the car
Giulia is going to visit a friend of hers for dinner, she’s late so she decides to use a PowerEnJoy car. While arriving to her friend house, because of the hurry for being late, she parks the car in the first parking she finds, she leaves the car and passes over the notification from the system that she’s not using a safe area parking on the satellite navigation screen and the acoustic signal from the car to catch her attention on the	

warning. A minute after, the system recognise that the car has been left in a wrong area so, it sends a notification to the nearest and free operator with the position of the car and the position where to leave it. In this whole time and also while the operator reaches the car and drives it to the safe area the system keeps charging Guilia as if she's riding the car herself.

3.1.1.12

Name	Re-charge on-site
Goal	[G9] Identifies correctly the duration of a ride. [G10] Rewards or punishes clients according to their behavior. [G11] Notifies operators when their intervention is needed.
Assumptions	1. Client is in the reserved car
Harry has almost finished his ride on a PowerEnJoy car and he is parking in a safe area. He turns off the engine and leaves the car without noticing that the remaining battery is less than 20%. After a minute the system applies 30% additional fee to Harry's ride, notifies the nearest operator to ask for his intervention and locks the car. The operator reaches the car and recharges it on site because on the app a message told him that the nearest power station is not reachable from that position with that percentage of battery.	

3.1.1.13

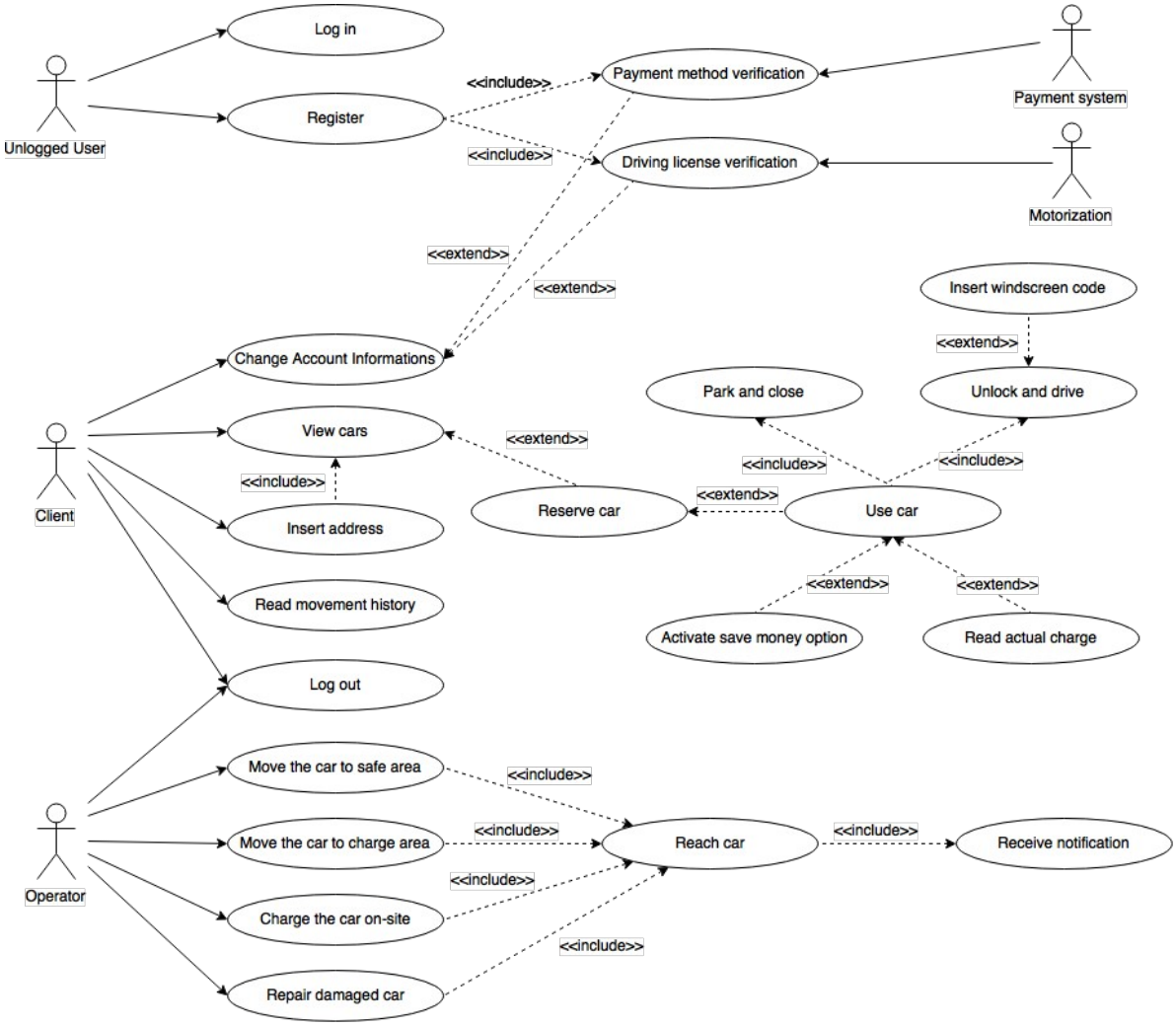
Name	Begin charging timer expires
Goal	[G3] Allows clients to find available cars. [G7] Charges the client according to the duration of the ride. [G8] Allows the clients to know the actual charge.
Assumptions	
Sarah has reserved a car to go shopping with her friend Anna, they agreed to meet each other next to the car a drive to the mall. Sarah reaches the car after 40 minutes from the reservation, but Anna is late, so Sarah decides to get in the car to avoid the reservation fee. After a minute the system recognize the time out from the timer and starts charging Sarah even if she hadn't switched on the car, showing her the charge on the screen. A few minutes later Anna reaches the car and the start their ride to the mall.	

3.1.1.14

Name	Damaged car
Goal	[G11] Notifies operators when their intervention is needed.
Assumptions	
Michele is an employee of PowerEnJoy. While working he receives a notification on the app that a car needs his intervention. Michele reaches the car which, as reported by the	

app, is broken but still drivable, so he brings it to a mechanic to do a revision. When the mechanic has finished the revision Michele takes the car and parks it in a safe area.

3.1.2 Use Case Diagram



3.1.2.1

Name	Login
Actors	Unlogged user
Entry conditions	• There are no entry conditions
Flow of events	• The unlogged user opens the application on the smartphone • The unlogged user inserts email and password • The unlogged user clicks on the ‘Login’ button
Exit conditions	• The system shows to the user his/her home view of the application
Exceptions	The user inserts wrong email and/or password so the system doesn’t let him/her login

3.1.2.2

Name	Register
Actors	Unlogged user, payment system, motorization
Entry conditions	● There are no entry conditions
Flow of events	● The unlogged user opens the application on the smartphone ● The unlogged user clicks on the register button ● The unlogged user fills the form ● The system verifies the payment method through a request to the payment system ● The system verifies the driving license through a request to the motorization ● The system sends the password to the user
Exit conditions	● The system shows a message of confirmed registration
Exceptions	The user inserts invalid email, payment method or driving license so the system doesn't let him/her register

3.1.2.3

Name	Change account informations
Actors	Client, payment system, motorization
Entry conditions	● The client has opened the application and is logged in
Flow of events	● The client taps on the menu button and selects change account informations ● The client modifies the informations and confirm ● If the payment method has been modified the system verifies it through a request to the payment system ● If the driving license has been modified the system verifies it through a request to the motorization ● The system applies the modifications
Exit conditions	● The system shows a confirmation message to the user
Exceptions	The user inserts invalid data, payment method or driving license so the system doesn't apply the modification

3.1.2.4

Name	View map
Actors	Client
Entry conditions	● The client has opened the application and is logged in

Flow of events	● The client sees the map shown by the system
Exit conditions	● The system shows the map
Exceptions	-

3.1.2.5

Name	Insert address
Actors	Client
Entry conditions	● The client has opened the application and is logged in
Flow of events	● The client inserts an address in the bar ● The system shows the map at the inserted address
Exit conditions	● The system shows the map at the inserted address
Exceptions	The client inserts an invalid address so the system doesn't refresh the map

3.1.2.6

Name	Reserve car
Actors	Client
Entry conditions	● The client has opened the application and is logged in ● The client sees the map
Flow of events	● The client selects an available car from the map ● The client confirms the reservation ● The system marks the car as reserved
Exit conditions	● The system shows a confirmation message
Exceptions	The client has already reserved a car so the system doesn't allow him/her to do multiple reservations

3.1.2.7

Name	Insert windscreen code
Actors	Client
Entry conditions	● The client has opened the application and is logged in ● The client is seeing the map ● The client is near the car
Flow of events	● The system requires the code shown on the windscreen of the

	car ● The client inserts the code
Exit conditions	● The client confirms the inserted code
Exceptions	-

3.1.2.8

Name	Unlock car and drive
Actors	Client
Entry conditions	● The client has opened the application and is logged in ● The client has done a reservation ● The client is near the car
Flow of events	● The client taps the ‘Unlock’ button in the app ● The system checks the GPS position of the client ● If the GPS is turned off the client **Insert windscreen code** (see use case) requested by the system ● The system unlocks the car and the client drives
Exit conditions	● The system marks the car as in use
Exceptions	The reservation is expired so the system doesn’t open the car The code inserted by the client is wrong so the system doesn’t unlock the car and asks it again

3.1.2.9

Name	Insert save money option
Actors	Client
Entry conditions	● The client is using the car
Flow of events	● The client taps the ‘Save money’ button on the screen of the car ● The system asks for a destination address ● The client inserts the destination address
Exit conditions	● The system shows the nearest charging station to the inserted address (ensuring uniform distribution of the cars)
Exceptions	The client inserts an invalid address so the system asks it again

3.1.2.10

Name	Read actual charge
Actors	Client

Entry conditions	• The client is using the car
Flow of events	• The client sees the actual charge on the screen of the car
Exit conditions	• The client sees the actual charge on the screen of the car
Exceptions	-

3.1.2.11

Name	Park and close
Actors	Client
Entry conditions	• The client is driving the car
Flow of events	• The client parks the car • The client exits the car • The client closes the car
Exit conditions	• The system locks the car
Exceptions	If the car isn't parked in a safe area the system shows a message on the screen of the car and activates a beep sound when the door is opened

3.1.2.12

Name	Use car
Actors	Client
Entry conditions	• The client has opened the application and is logged in • The client has done a reservation • The client is near the car
Flow of events	• The client uses the app to **Unlock and drive** the car • The client, once finished the ride, **Park and close** the car
Exit conditions	• The system marks the car as available
Exceptions	-

3.1.2.13

Name	Read movement history
Actors	Client
Entry conditions	• The client has opened the application and is logged in
Flow of events	• The client uses the app to access the payments section

	● The client sees all the executed payments ● The client selects one payment
Exit conditions	● The system shows the details of the selected payment
Exceptions	The client doesn't have any payment to show

3.1.2.14

Name	Logout
Actors	Client Operator
Entry conditions	● The user has opened the application and is logged in
Flow of events	● The user taps the logout button
Exit conditions	● The system shows the initial view of the application for non logged users
Exceptions	-

3.1.2.15

Name	Move the car to safe area
Actors	Operator
Entry conditions	● The operator is logged in ● The car is in an unsafe area
Flow of events	● The operator receives a notification of a car parked in an unsafe area ● The operator reaches the car ● The operator enters in the car ● The operator moves the car and parks it in a safe area
Exit conditions	● The operator exits the car
Exceptions	-

3.1.2.16

Name	Move the car to charge area
Actors	Operator
Entry conditions	● The operator is logged in ● The car has a low battery level ● The car is near enough a charging station

Flow of events	● The operator receives a notification of a car that must be moved in a charging station ● The operator reaches the car ● The operator enters in the car ● The operator moves the car and plugs it in a charging station
Exit conditions	● The operator exits the car
Exceptions	-

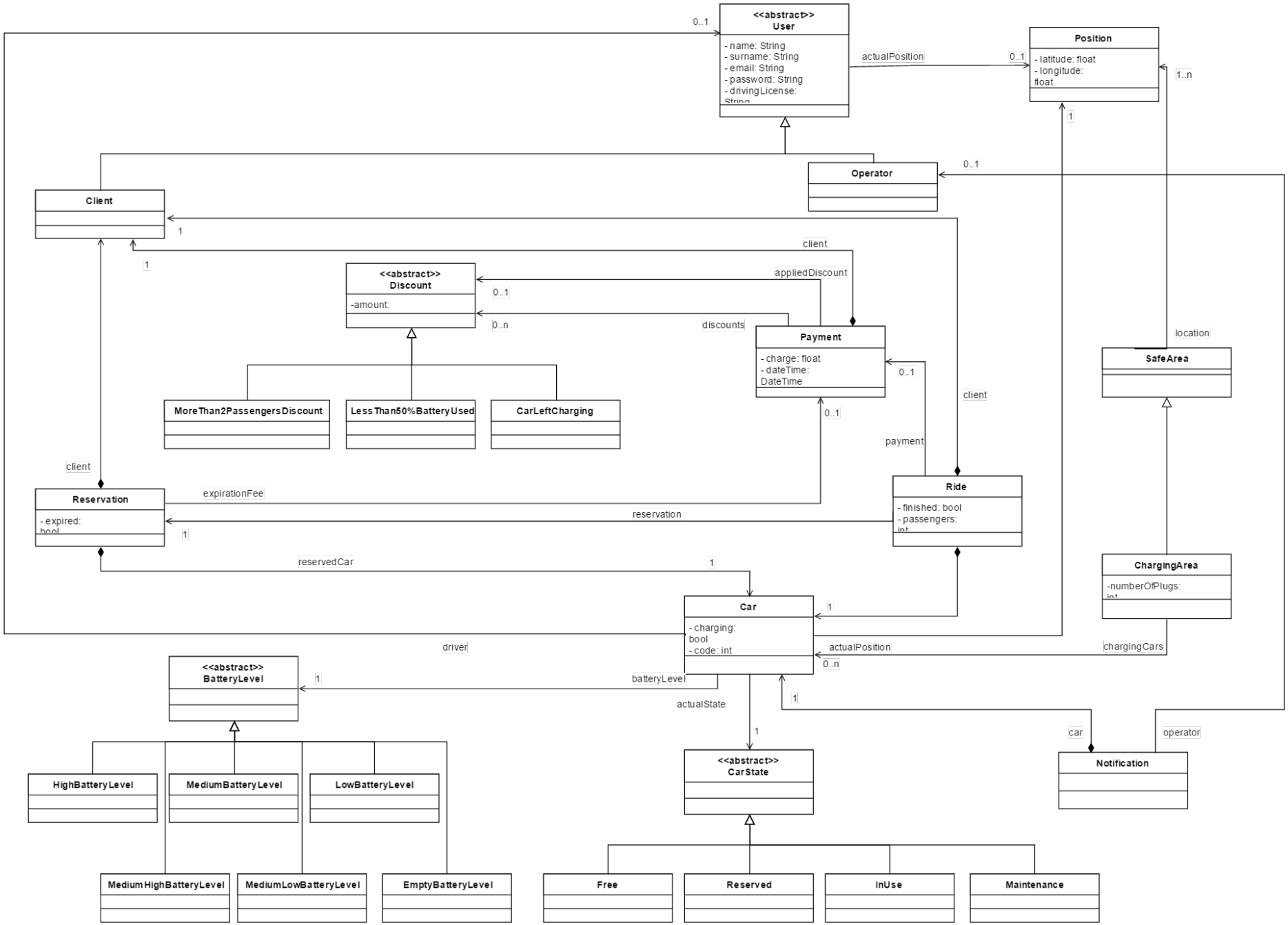
3.1.2.17

Name	Charge the car on-site
Actors	Operator
Entry conditions	● The operator is logged in ● The car has a low battery level ● The car is too far from the nearest charging area
Flow of events	● The operator receives a notification of a car that must be charged on-site ● The operator reaches the car ● The operator charges the car
Exit conditions	● The car has enough charge
Exceptions	-

3.1.2.18

Name	Repair damaged car
Actors	Operator
Entry conditions	● The operator is logged in ● The car is damaged
Flow of events	● The operator receives a notification of a car is damaged ● The operator reaches the car ● The operator takes care of the car according to the situation ● The operator relocates the car in a safe area (if it has been moved)
Exit conditions	● The car is repaired and ready to be used
Exceptions	-

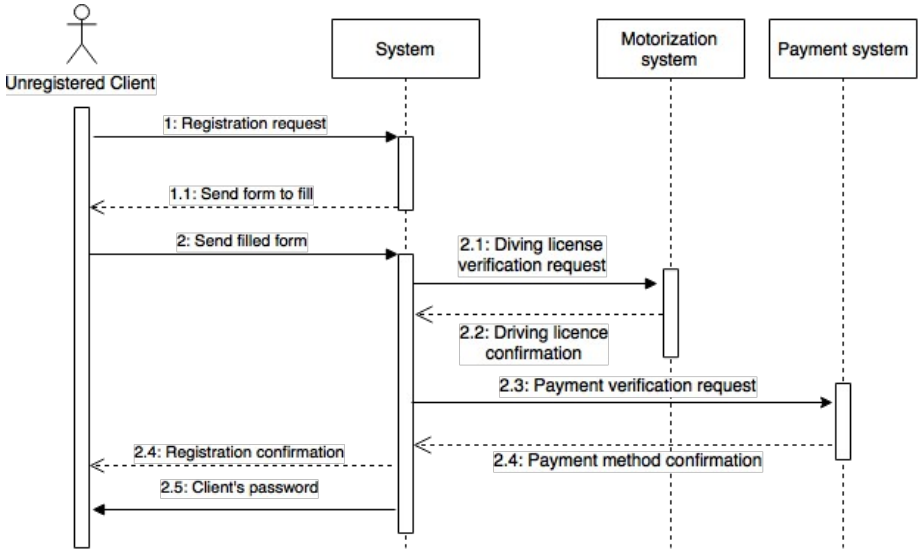
3.1.3 Analysis Model



3.1.4 Sequence Diagram

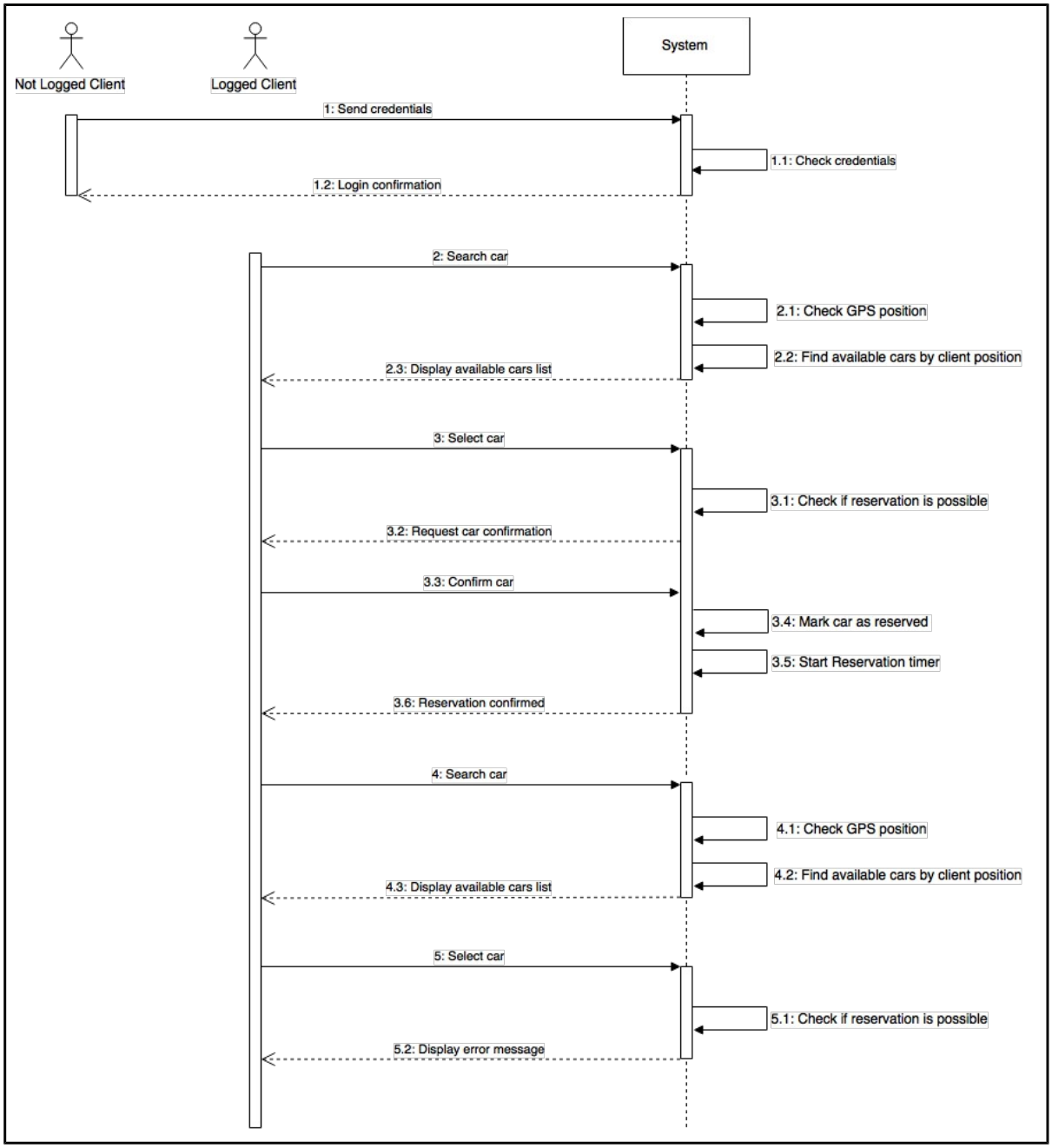
3.1.4.1

Register into the system	
Actors	Unregistered client
Flow of events	1. The unregistered user sends the registration request 2. The unregistered user fills the form 3. The system checks the driving license, the payment method and confirm the registration 4. The system gives a password to the new client



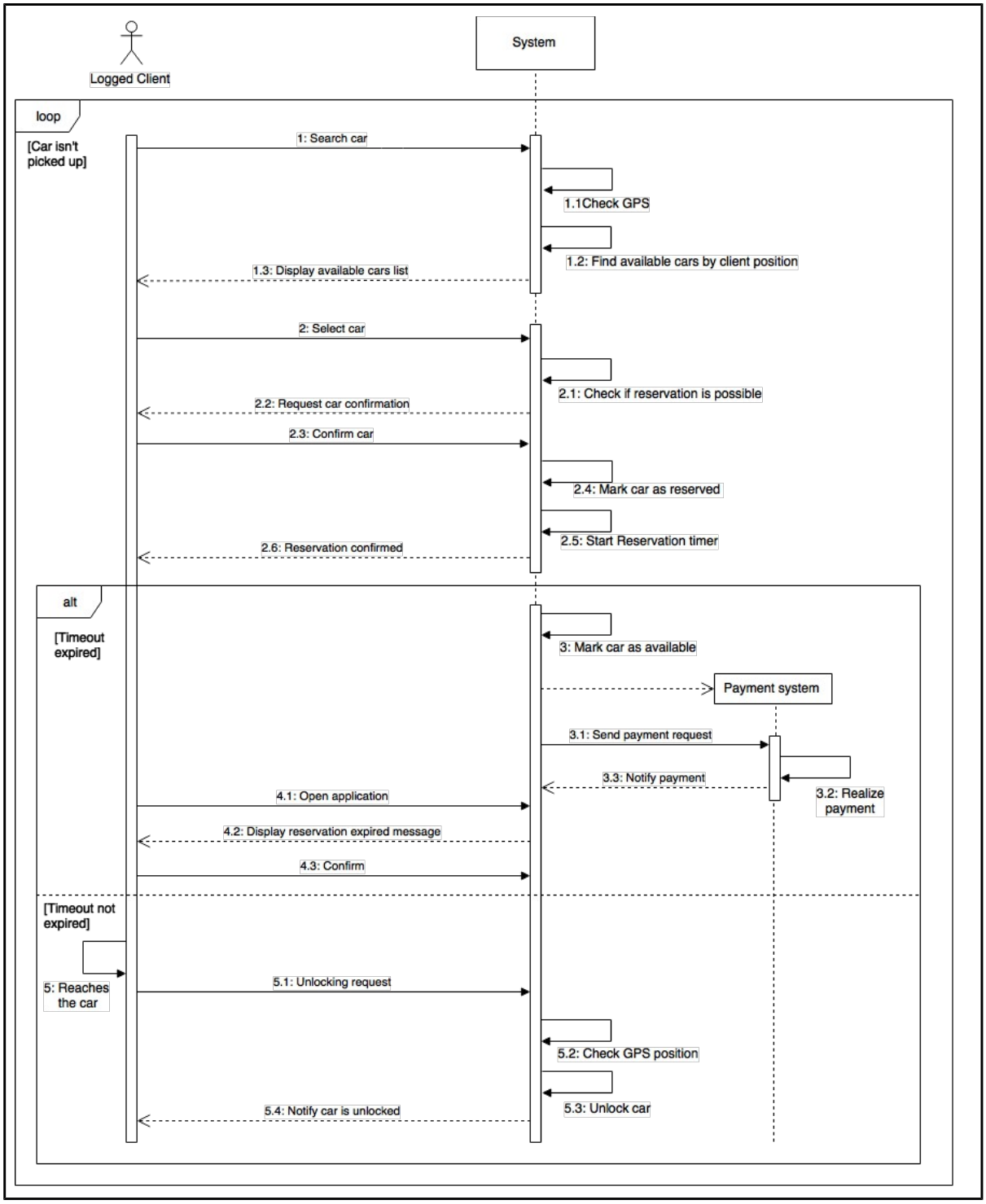
3.1.4.2

Login the system and double reservation	
Actors	Not logged client Logged client
Flow of events	1. Not logged client sends credentials to login 2. Logged client searches for a car near his/her position 3. Logged client reserves a car 4. Logged client searches again for a car near his/her position 5. Logged client reserves another car 6. The system displays an error message



3.1.4.3

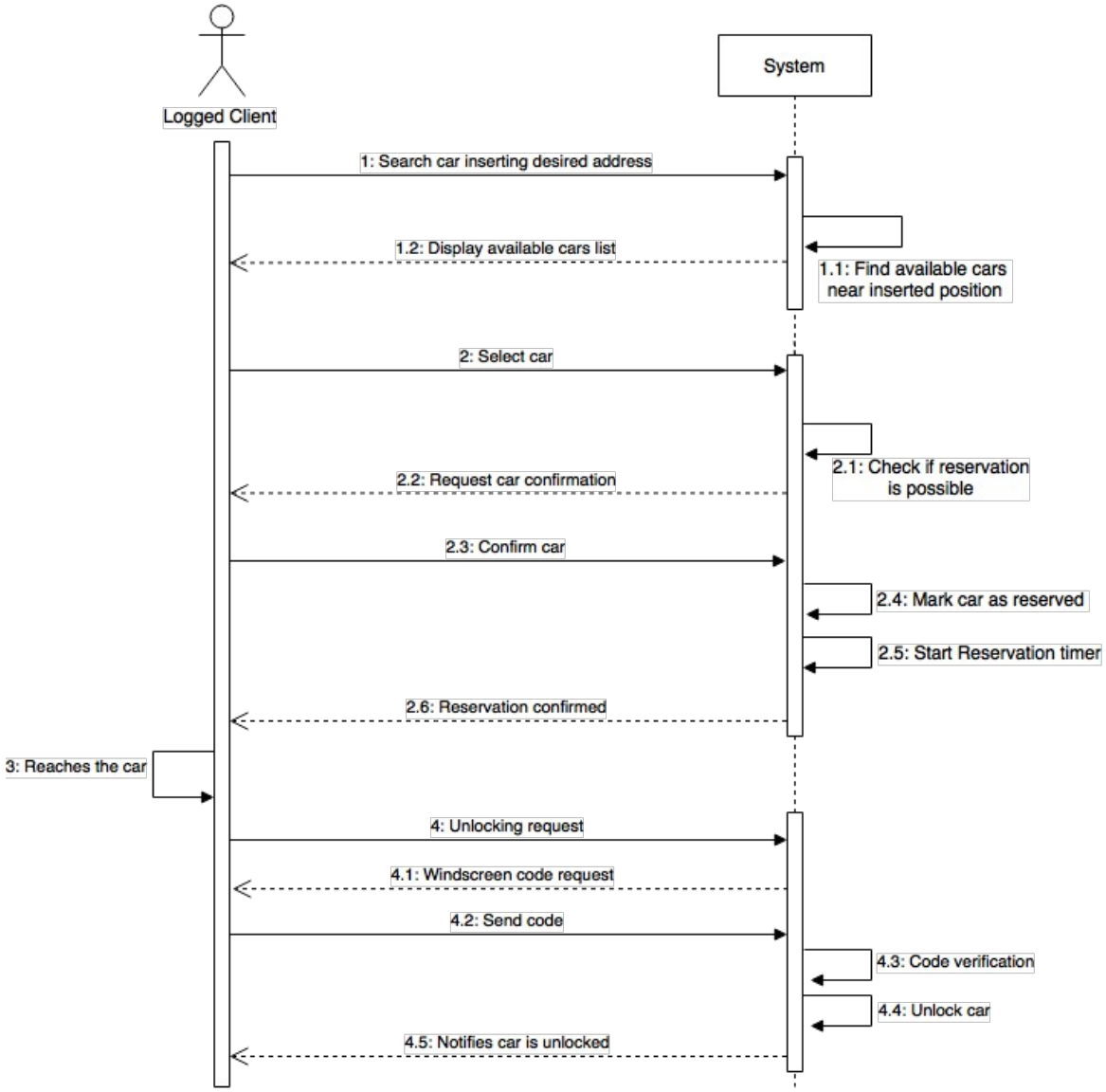
Expiration of the reservation	
Actors	Logged client
Flow of events	1. Logged client searches for a car near his/her position 2. Logged client reserves a car 3. Reservation timeout expires 4. The system notifies the expiration 5. Logged client searches for a car near his/her position 6. Logged client reserves a car 7. Logged client reaches the car and request the unlock 8. The system verifies the client position and unlocks the car



3.1.4.4

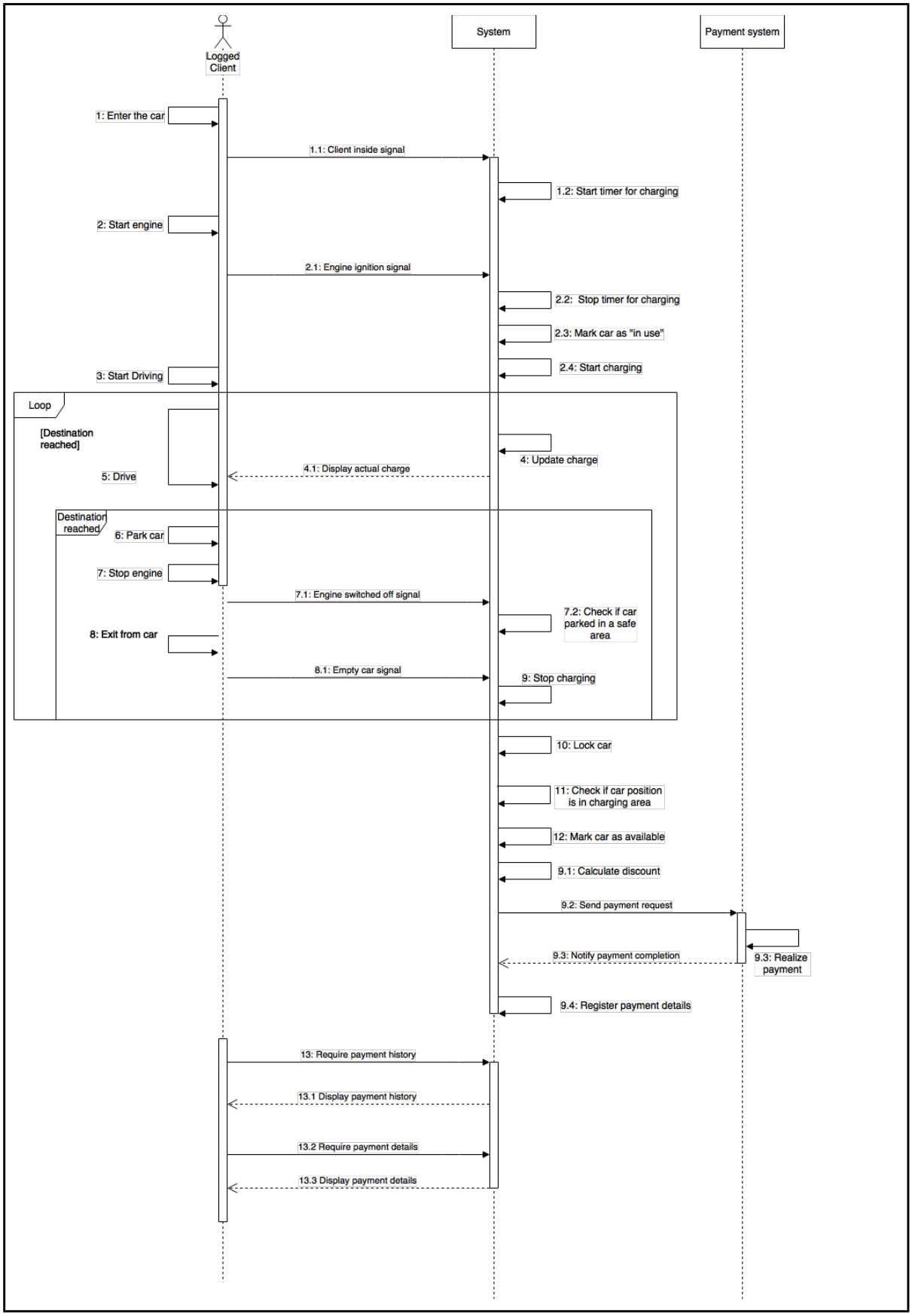
Searching through address and car unlocking	
Actors	Logged client
Flow of events	1. Logged client inserts the desired address 2. Logged client reserves a car 3. Logged client reaches the car and request the unlock

	4. Logged client sends the widescreen code 5. The system checks the code and unlocks the car



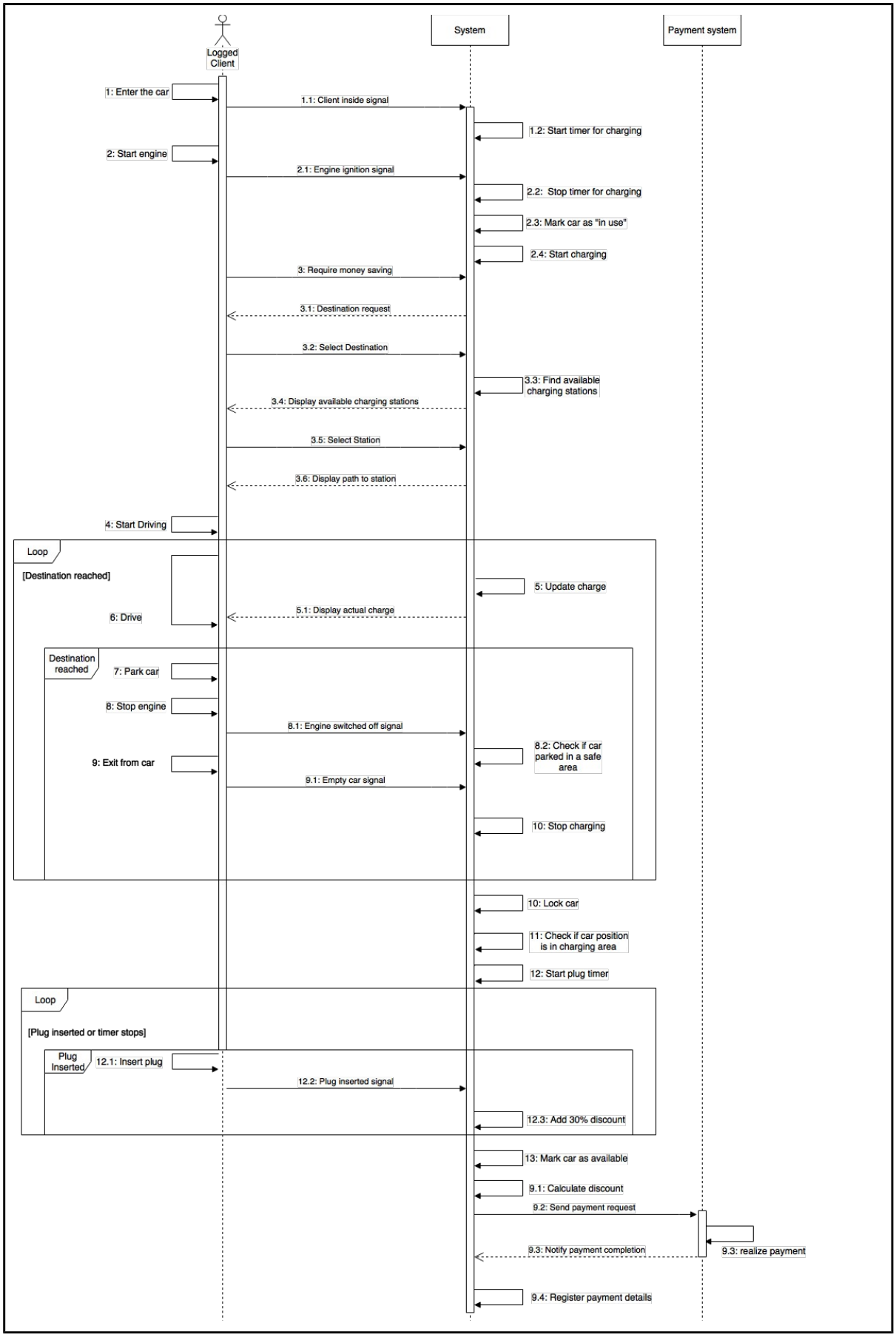
3.1.4.5

Ride and payment	
Actors	Logged client
Flow of events	1. Logged client enters the car, ignites the engine and drives 2. The system updates the charge and shows it 3. Logged client parks the car, switches off the engine and exits 4. The system locks the car, calculates any discount, sends a payment request and registers it 5. The client requires payment history and details


Logged Client
System
Payment system
1: Enter the car
1.1: Client inside signal
1.2: Start timer for charging
2: Start engine
2.1: Engine ignition signal
2.2: Stop timer for charging
2.3: Mark car as "in use"
2.4: Start charging
3: Start Driving
Loop
[Destination reached]
4: Update charge
4.1: Display actual charge
5: Drive
Destination reached
6: Park car
7: Stop engine
7.1: Engine switched off signal
7.2: Check if car parked in a safe area
8: Exit from car
8.1: Empty car signal
9: Stop charging
10: Lock car
11: Check if car position is in charging area
12: Mark car as available
9.1: Calculate discount
9.2: Send payment request
9.3: Realize payment
9.3: Notify payment completion
9.4: Register payment details
13: Require payment history
13.1 Display payment history
13.2 Require payment details
13.3 Display payment details

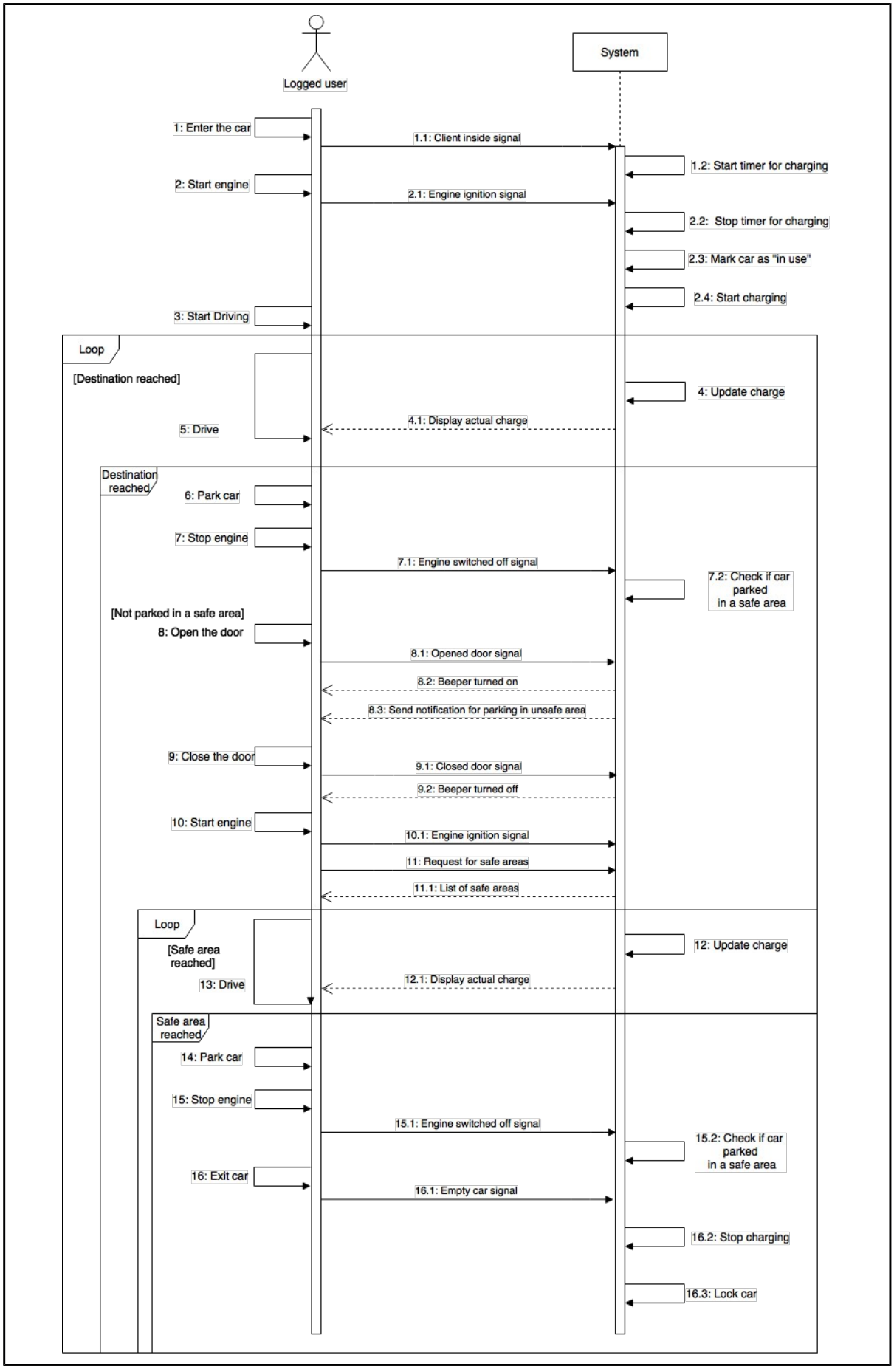
3.1.4.6

Money saving	
Actors	Logged client
Flow of events	1. Logged client enters the car, starts the engine and activate save money option 2. Logged client insert his/her destination 3. Logged client selects a power station 4. Logged client drives, parks and exit the car 5. Logged client inserts the plug 6. The system calculates the discount and registers the payment


Logged Client
System
Payment system
1: Enter the car
1.1: Client inside signal
1.2: Start timer for charging
2: Start engine
2.1: Engine ignition signal
2.2: Stop timer for charging
2.3: Mark car as "in use"
2.4: Start charging
3: Require money saving
3.1: Destination request
3.2: Select Destination
3.3: Find available charging stations
3.4: Display available charging stations
3.5: Select Station
3.6: Display path to station
4: Start Driving
Loop
[Destination reached]
5: Update charge
6: Drive
5.1: Display actual charge
Destination reached
7: Park car
8: Stop engine
8.1: Engine switched off signal
8.2: Check if car parked in a safe area
9: Exit from car
9.1: Empty car signal
10: Stop charging
10: Lock car
11: Check if car position is in charging area
12: Start plug timer
Loop
[Plug inserted or timer stops]
Plug Inserted
12.1: Insert plug
12.2: Plug inserted signal
12.3: Add 30% discount
13: Mark car as available
9.1: Calculate discount
9.2: Send payment request
9.3: realize payment
9.3: Notify payment completion
9.4: Register payment details

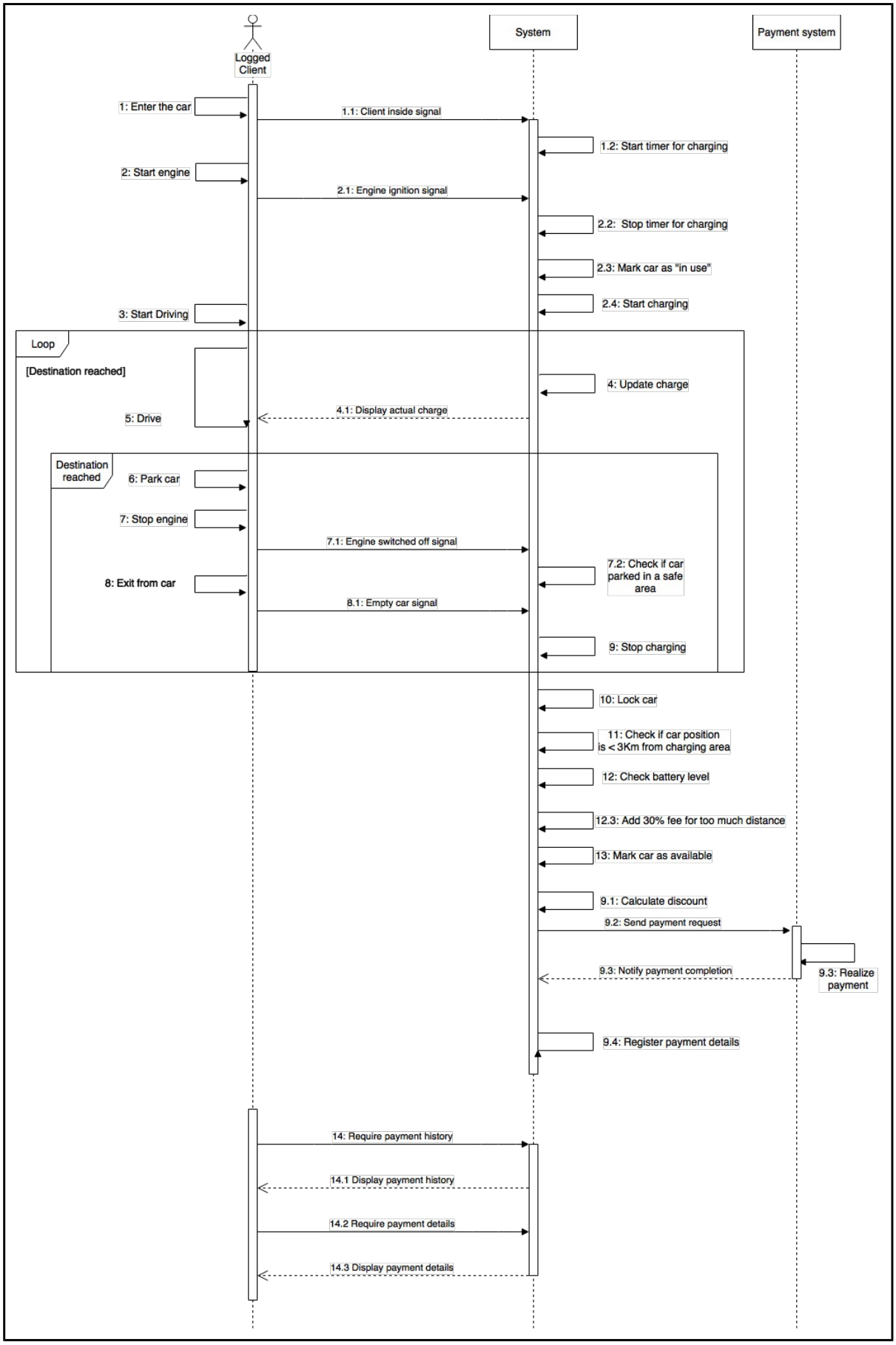
3.1.4.7

Unsafe area parking	
Actors	Logged client
Flow of events	1. Logged client enters the car, starts the engine and drives 2. Logged client reaches his/her destination, parks in an unsafe area and opens the door to exit 3. The system activates a beeper 4. Logged client closes the door 5. The system stops the beeper 6. Logged client drives, reaches a safe area, parks and exits the car 7. The system locks the car and stops the charge


Logged user
System
1: Enter the car
1.1: Client inside signal
1.2: Start timer for charging
2: Start engine
2.1: Engine ignition signal
2.2: Stop timer for charging
2.3: Mark car as "in use"
2.4: Start charging
3: Start Driving
Loop
[Destination reached]
4: Update charge
5: Drive
4.1: Display actual charge
Destination reached
6: Park car
7: Stop engine
7.1: Engine switched off signal
7.2: Check if car parked in a safe area
[Not parked in a safe area]
8: Open the door
8.1: Opened door signal
8.2: Beeper turned on
8.3: Send notification for parking in unsafe area
9: Close the door
9.1: Closed door signal
9.2: Beeper turned off
10: Start engine
10.1: Engine ignition signal
11: Request for safe areas
11.1: List of safe areas
Loop
[Safe area reached]
12: Update charge
13: Drive
12.1: Display actual charge
Safe area reached
14: Park car
15: Stop engine
15.1: Engine switched off signal
15.2: Check if car parked in a safe area
16: Exit car
16.1: Empty car signal
16.2: Stop charging
16.3: Lock car

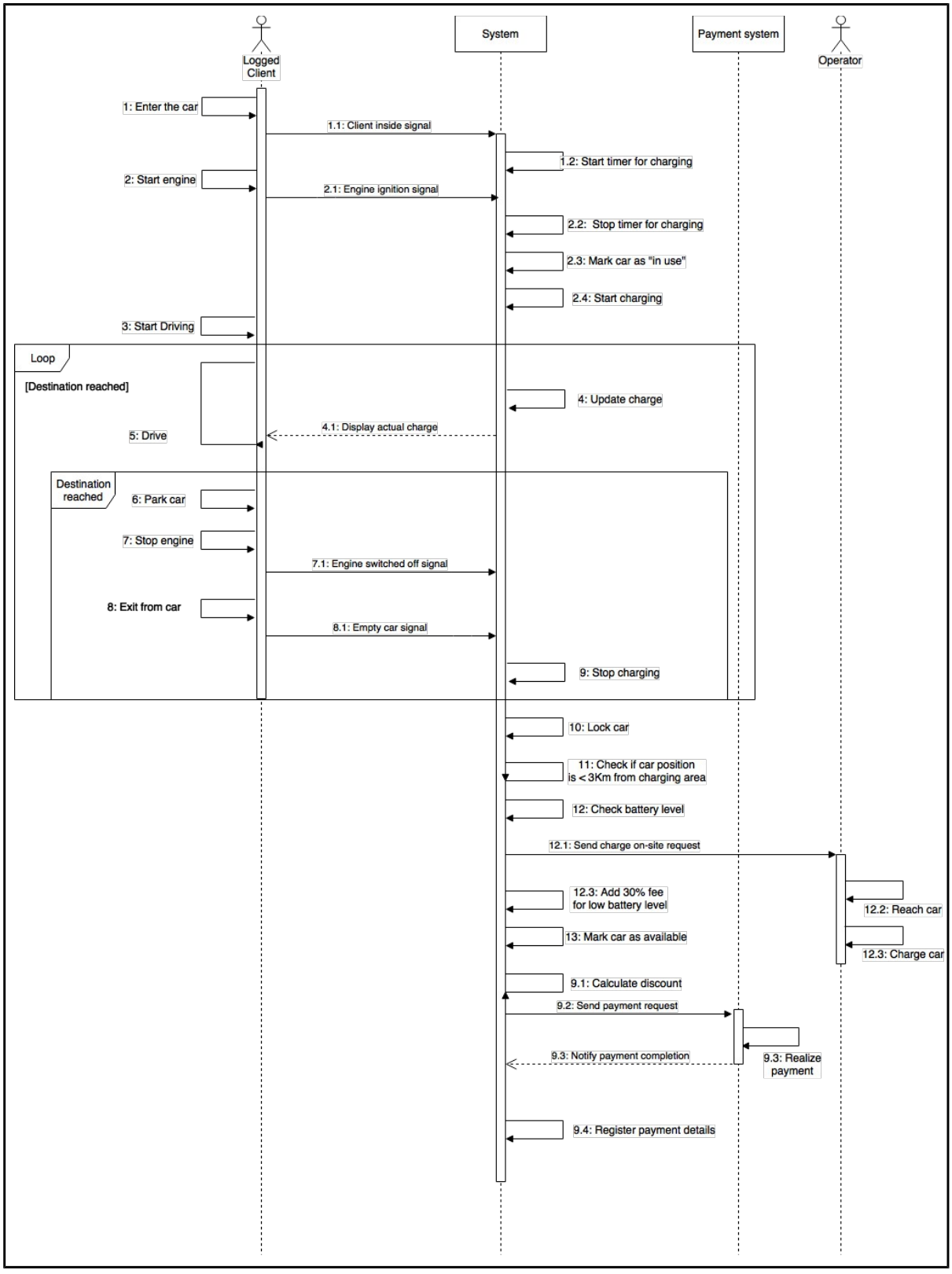
3.1.4.8

Bad behaviour: low battery	
Actors	Logged client
Flow of events	1. Logged client enters the car, starts the engine and starts driving 2. Logged client drives until he/she parks the car in a safe area 3. Logged client exits the car 4. The system locks the car and checks the distance from the nearest charging station 5. The system adds 30% extra charge for too much distance from charging station and sends the payment request 6. Logged client request payment history 7. Logged client request details of the last payment


Logged Client
System
Payment system
1: Enter the car
1.1: Client inside signal
1.2: Start timer for charging
2: Start engine
2.1: Engine ignition signal
2.2: Stop timer for charging
2.3: Mark car as "in use"
2.4: Start charging
3: Start Driving
Loop
[Destination reached]
4: Update charge
4.1: Display actual charge
5: Drive
Destination reached
6: Park car
7: Stop engine
7.1: Engine switched off signal
7.2: Check if car parked in a safe area
8: Exit from car
8.1: Empty car signal
9: Stop charging
10: Lock car
11: Check if car position is < 3Km from charging area
12: Check battery level
12.3: Add 30% fee for too much distance
13: Mark car as available
9.1: Calculate discount
9.2: Send payment request
9.3: Realize payment
9.3: Notify payment completion
9.4: Register payment details
14: Require payment history
14.1 Display payment history
14.2 Require payment details
14.3 Display payment details

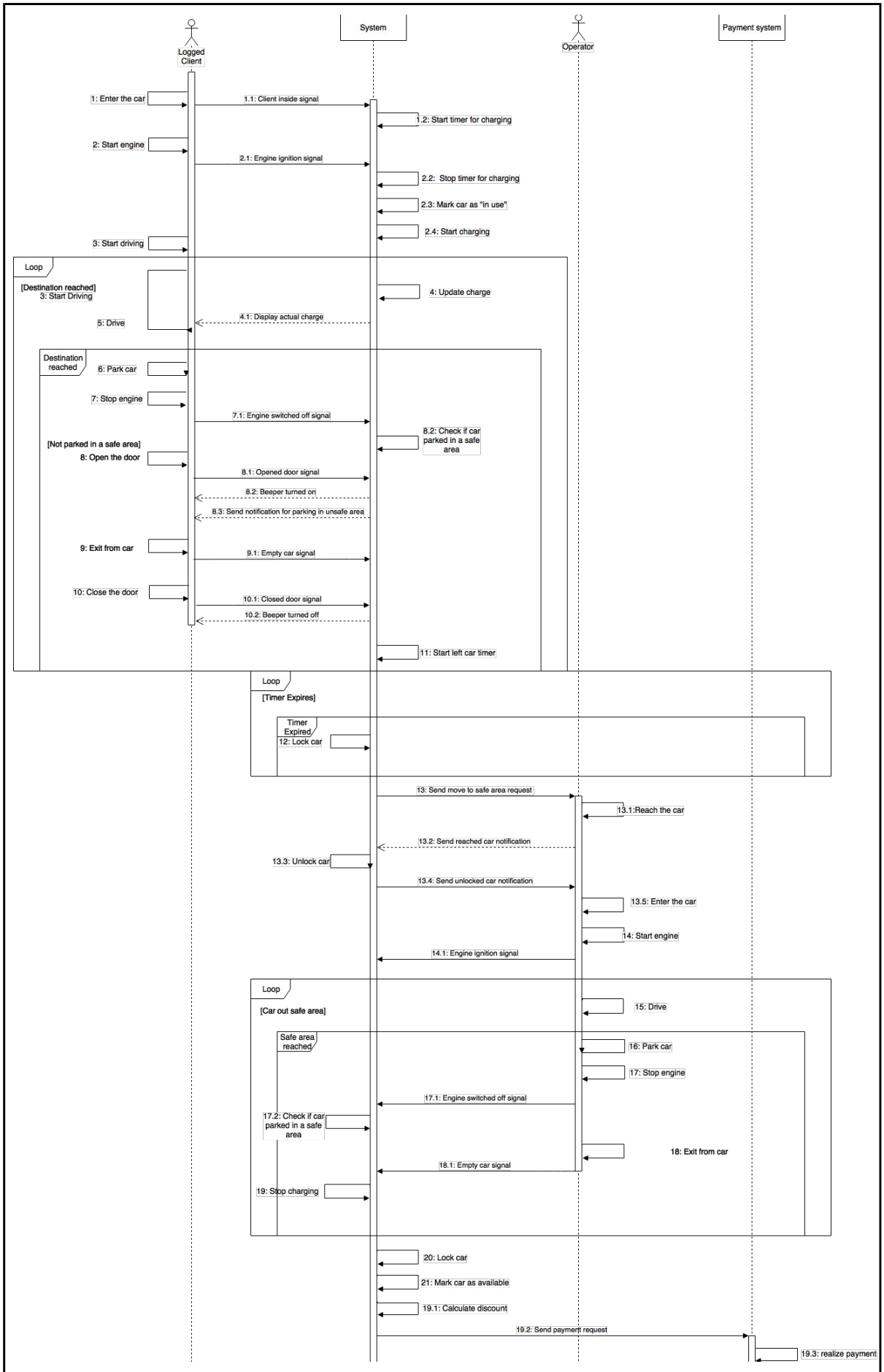
3.1.4.9

Re-charge on-site	
Actors	Logged client Operator
Flow of events	1. Logged client enters the car, starts the engine and starts driving 2. Logged client drives until he/she parks the car in a safe area 3. Logged client exits the car 4. The system locks the car and checks battery level 5. The system sends a request of charge on-site to the operator 6. The operator reaches the car and charges it 7. The system adds 30% extra charge for low battery level and sends the payment request 8. The system registers the payment request


Logged Client
System
Payment system
Operator
1: Enter the car
1.1: Client inside signal
1.2: Start timer for charging
2: Start engine
2.1: Engine ignition signal
2.2: Stop timer for charging
2.3: Mark car as "in use"
2.4: Start charging
3: Start Driving
Loop
[Destination reached]
4: Update charge
4.1: Display actual charge
5: Drive
Destination reached
6: Park car
7: Stop engine
7.1: Engine switched off signal
8: Exit from car
8.1: Empty car signal
9: Stop charging
10: Lock car
11: Check if car position is < 3Km from charging area
12: Check battery level
12.1: Send charge on-site request
12.3: Add 30% fee for low battery level
12.2: Reach car
13: Mark car as available
12.3: Charge car
9.1: Calculate discount
9.2: Send payment request
9.3: Notify payment completion
9.3: Realize payment
9.4: Register payment details

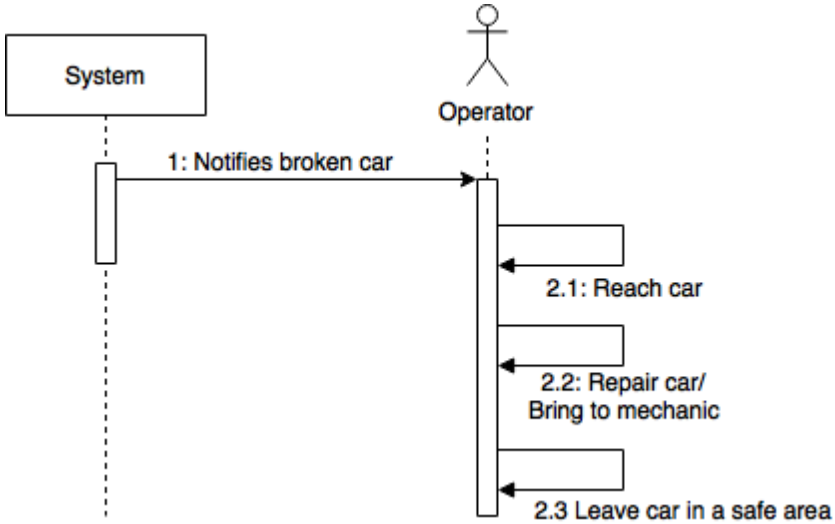
3.1.4.10

Operator relocates car correctly	
Actors	Logged client, Operator
Flow Of Events	1. Logged client enters the car, starts the engine and drives 2. Logged client reaches his/her destination, parks in an unsafe area and opens the door to exit 3. The system activates a beeper 4. Logged client ignores the warnings and leave the car 5. The system stops the beeper 6. The system starts the timer to wait client 7. The system notifies the nearest operator to move the car 8. Operator reaches the car 9. Operator drives the car to a safe area 10. Operator leaves the car 11. The system stops the charging 12. The system puts the car available back 13. The system calculates eventual discount 14. The system uses the payment system to charge effectively the client


Logged Client
System
Operator
Payment system
1: Enter the car
1.1: Client inside signal
1.2: Start timer for charging
2: Start engine
2.1: Engine ignition signal
2.2: Stop timer for charging
2.3: Mark car as "in use"
2.4: Start charging
3: Start driving
Loop
[Destination reached]
3: Start Driving
4: Update charge
4.1: Display actual charge
5: Drive
Destination reached
6: Park car
7: Stop engine
7.1: Engine switched off signal
8.2: Check if car parked in a safe area
[Not parked in a safe area]
8: Open the door
8.1: Opened door signal
8.2: Beeper turned on
8.3: Send notification for parking in unsafe area
9: Exit from car
9.1: Empty car signal
10: Close the door
10.1: Closed door signal
10.2: Beeper turned off
11: Start left car timer
Loop
[Timer Expires]
Timer Expired
12: Lock car
13: Send move to safe area request
13.1:Reach the car
13.2: Send reached car notification
13.3: Unlock car
13.4: Send unlocked car notification
13.5: Enter the car
14: Start engine
14.1: Engine ignition signal
Loop
[Car out safe area]
15: Drive
Safe area reached
16: Park car
17: Stop engine
17.1: Engine switched off signal
17.2: Check if car parked in a safe area
18: Exit from car
18.1: Empty car signal
19: Stop charging
20: Lock car
21: Mark car as available
19.1: Calculate discount
19.2: Send payment request
19.3: realize payment

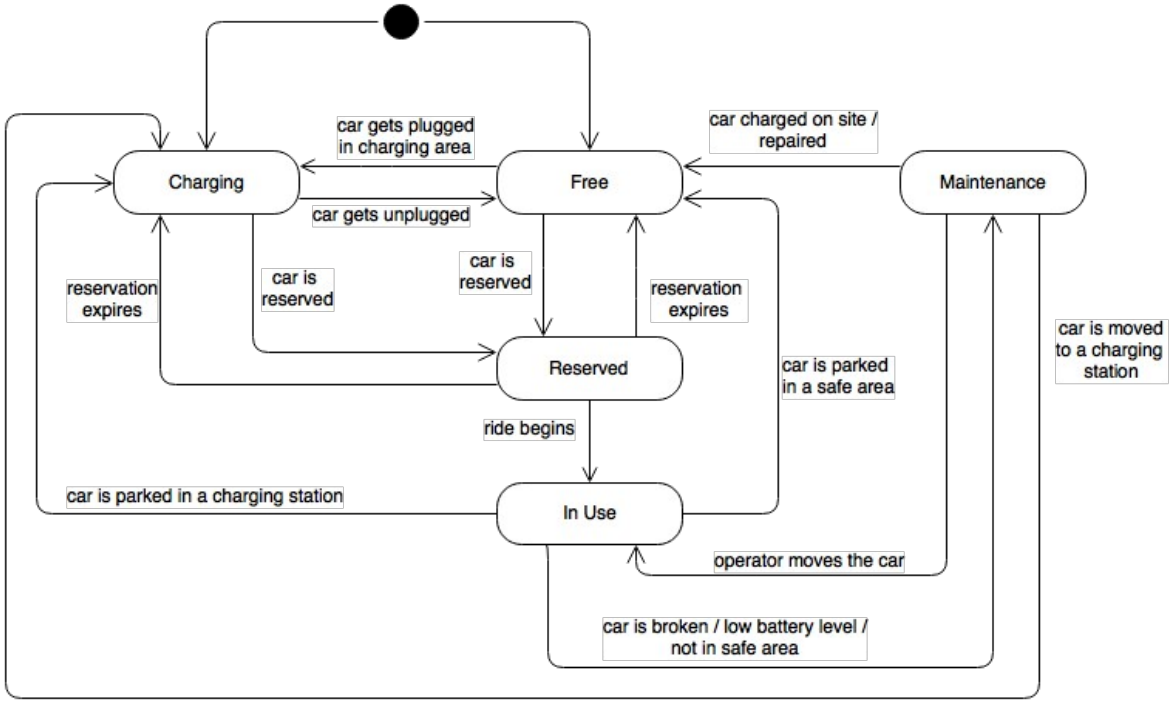
3.1.4.11

Repair of a damaged car	
Actors	Operator
Flow of events	1. A car needs to be repaired 2. The system notifies the nearest operator 3. The operator reaches the car 4. The operator repairs the car or brings it where it can be repaired (according to the situation) 5. The operator leaves the car in a safe area



3.1.5 State Chart Diagram

3.1.5.1 Car State Chart Diagram



3.2 Non-Functional requirements

- The system find and sort available car for clients in a reasonable amount of time (for this specific requirement a reasonable time is under one minute).
- The notified operator reaches the car in reasonable amount of time (for this specific requirement a reasonable time is under half an hour).
- The server must be available 24 hours per day.
- The satellite navigator always find the best way to reach the destination.
- Tickets will be charged to the responsible driver

4 Alloy

4.1 Code

```
open util/boolean

sig Position {}

pred InsideArea [car: Car, safeArea: SafeArea]
{
	car.actualPosition in safeArea.positions
}

sig UnregisteredUser {}

abstract sig User
{
	email: one Email,
	actualPosition: one Position
}

sig Email {}

fact oneMailPerUser
{
	all e: Email | one user : User | user.email = e
}

fact noUserInSamePosition
{
	all disjoint u1, u2: User | u1.actualPosition != u2.actualPosition
}

fact emailIsUnique
{
	all disjoint u1,u2: User | u1.email != u2.email
}

sig Client extends User {}

sig Operator extends User {}

sig Car
{
	charging: one Bool,
	driver: lone User,
	actualPosition: one Position,
	code: one Code,
	batteryLevel: one BatteryLevel,
	state: one CarState
```

```
}

sig Code {}

fact everyCodeAssignedToCar
{
	all c: Code | one car: Car | car.code = c
}

abstract sig BatteryLevel {}

one sig HighBatteryLevel extends BatteryLevel {}

one sig MediumHighBatteryLevel extends BatteryLevel {}

one sig MediumLowBatteryLevel extends BatteryLevel {}

one sig LowBatteryLevel extends BatteryLevel {}

one sig EmptyBatteryLevel extends BatteryLevel{}

fact atMostOneCarForDriver
{
	all disjoint c1, c2 : Car | ( c1.driver != none and c2.driver != none ) implies c1.driver !=
c2.driver
}

fact noCarsInSamePosition
{
	all disjoint c1, c2: Car | c1.actualPosition != c2.actualPosition
}

abstract sig CarState {}

one sig Free extends CarState {}

one sig Reserved extends CarState {}

one sig InUse extends CarState {}

one sig Maintenance extends CarState{}

fact carNotReservableDuringMaintenance
{
	all car:Car | car.state = Maintenance implies no re : Reservation | re.reservedCar = car
}

fact carNotDrivableDuringMaintenance
{
	all car:Car | ( car.state = Maintenance ) implies
	( ( car.driver = none ) and
```

```
		( ( car.batteryLevel = LowBatteryLevel or car.batteryLevel = EmptyBatteryLevel ) ) and
		( one safeArea : SafeArea | InsideArea[car, safeArea] ) )
}

fact carInMaintenanceOutOfChargingArea
{
		all c: Car | c.state = Maintenance implies no ca: ChargingArea | c.actualPosition in
ca.positions
}

fact chargingConditions
{
		all car: Car | car.charging = True implies (car.state = Free or car.state = Reserved
		or car.state = Maintenance)
}

fact noUserWhileFreeOrReserved
{
		all c: Car | (c.state = Free or c.state = Reserved) implies c.driver = none
}

fact driverInsideWhileDriving
{
		all c: Car | c.state = InUse implies ( c.driver != none and
		c.actualPosition = c.driver.actualPosition and
		( c.driver != Operator implies one re : Reservation | re.reservedCar = c and re.client =
c.driver ) )
}

fact codesOfTheCarsAreUnique
{
		all c1, c2: Car | (c1!=c2)=>c1.code!=c2.code
}

fact carStateInSafeArea
{
		all car: Car | (car.state = Free or car.state = Reserved) implies
		some safeArea: SafeArea | InsideArea[car, safeArea]
}

fact noEnergyLawViolation
{
		all car: Car | car.batteryLevel = EmptyBatteryLevel implies
		(car.state != InUse and car.state != Reserved)
}

sig Notification
{
		operator: lone Operator,
		car: one Car
}
```

```
fact oneNotificationPerOperatorAndCar
{
	all disjoint n1, n2: Notification | n1.operator != n2.operator and n1.car != n2.car
}

fact notificationOnlyWhenNeeded
{
	all c: Car | ((c.driver in Client and c.driver != none) or (c.state = Free) or (c.state = Reserved))
	implies (no n: Notification | n.car = c)
}

fact operatorNotifiedWhenDrives
{
	all c: Car | (c.driver in Operator and c.driver != none) implies (one n: Notification | n.car = c
and	n.operator = c.driver)
}

fact operatorNotifiedForManteinance
{
	all c: Car | c.state = Maintenance implies one n: Notification | n.car = c
}

fact nearestOperator
{
	all n: Notification | n.car.actualPosition != n.operator.actualPosition implies no o:
	Operator | o.actualPosition = n.car.actualPosition
}

fact operatorInSameCarPositionForChargeOnSite
{
	(all n: Notification | n.car.charging = True implies n.operator.actualPosition =
		n.car.actualPosition) and
	(all n: Notification | n.operator.actualPosition = n.car.actualPosition and n.car.state =
	Maintenance implies n.car.charging = True)
}

fact carsInMaintenanceIfLowBattery
{
	all car : Car | ( ( ( car.batteryLevel = LowBatteryLevel or car.batteryLevel =
EmptyBatteryLevel)
	and ( car.charging = False ) ) and
	( one safeArea : SafeArea | InsideArea[car, safeArea] ) and
	( car.driver = none ) ) implies car.state = Maintenance
}

pred FreeOperator[ o : Operator]
{
	no n : Notification | n.operator = o
}

fact pendingNotifications
{
```

```
        all n : Notification | ( ( n.operator != none ) or ( n.operator = none and  ( no o : Operator |
        FreeOperator[ o ] ) ) ) and
        not (  ( n.operator != none ) and ( n.operator = none and  ( no o : Operator |
FreeOperator[ o ] ) )  )
}

sig SafeArea
{
        positions: set Position
}
{
        #positions > 0
}

fact noSharedPositions
{
        all disjoint sa1, sa2: SafeArea | sa1.positions & sa2.positions = none
}

sig ChargingArea extends SafeArea
{
        numberOfPlugs: one Int,
        chargingCars: set Car
}
{
        numberOfPlugs > 0 and
        numberOfPlugs <= #positions and
        #chargingCars <= numberOfPlugs
}

fact chargingCarsAreInTheChargingArea
{
        all car : Car, chargingArea : ChargingArea | car in chargingArea.chargingCars implies
        InsideArea [car, chargingArea]
}

fact chargingCarsInChargingAreaOrMaintenanceState
{
        all c: Car | c.charging = True implies ( ( ( one ca: ChargingArea | c in ca.chargingCars ) or
        c.state = Maintenance ) and not
        ( ( one ca: ChargingArea | c in ca.chargingCars ) and c.state = Maintenance ) )
}

sig Reservation
{
        client: one Client,
        reservedCar: lone Car,
        expirationFee: lone Payment,
        expired: one Bool
}
{
        ( expired = True implies ( reservedCar = none and expirationFee != none) ) and
```

```
	( expired = False implies ( expirationFee = none  ) )
}

fact oneReservationPerCar
{
	all disjoint r1, r2: Reservation | ( r1.expired = False and r2.expired = False ) implies
	r1.reservedCar != r2.reservedCar
}

fact oneActiveReservationPerClient
{
	all disjoint r1, r2 : Reservation | ( r1.expired = False and r2.expired = False ) implies r1.client !
=
	r2.client
}

fact noReservationWithOutOfBatteryCars
{
	all r: Reservation | r.reservedCar.batteryLevel != LowBatteryLevel and
	r.reservedCar.batteryLevel != EmptyBatteryLevel
}

sig Ride
{
	client: one Client,
	reservation: one Reservation,
	passengers: one Int,
	payment: lone Payment,
	finished: one Bool
}
{
	( passengers >= 0 and passengers <= 4 ) and
	( finished = True implies ( payment != none and reservation.reservedCar = none and
	reservation.expired = False ) ) and
	( finished = False implies ( payment = none and reservation.reservedCar != none and
	reservation.expired = False  ) )
}

fact atMostOneRideForReservation
{
	all disjoint r1, r2: Ride | r1.reservation != r2.reservation
}

pred relatedRideExists [re : Reservation]
{
	one ri : Ride | ri.reservation = re
}

fact sameRiderThatReserved
{
	all ri : Ride | ri.reservation.client = ri.client
}
```

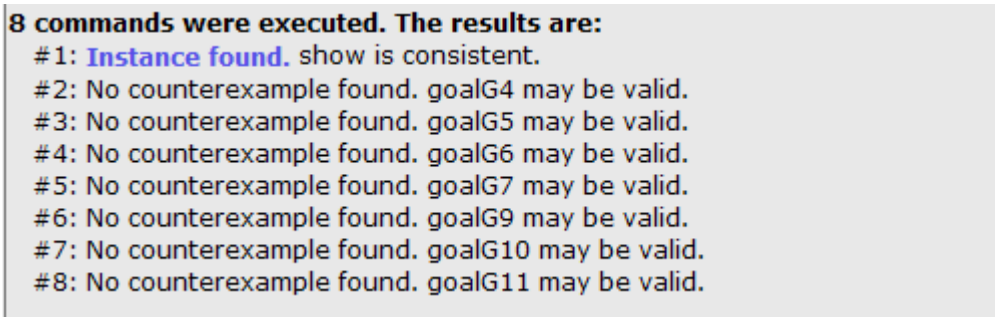
```
fact carStateWhileReserved
{
	( all c : Car | c.state = Reserved implies one re : Reservation | re.reservedCar = c and not
	relatedRideExists[re] ) and
	( all re : Reservation | ( not relatedRideExists[re] and re.expired = False ) implies one c : Car |
	re.reservedCar = c )
}

fact carStateWhileInUse
{
	( all ri : Ride | ri.finished = False implies ri.reservation.reservedCar.state = InUse )
}

fact existsRideOrReservedCarHasNotBeenPickedUpYet
{
	all re : Reservation | ( re.expired = False ) implies (  ( re.reservedCar.state = Reserved or
( one r
	: Ride | ( r.reservation = re ) ) ) and not ( re.reservedCar.state = Reserved and ( one r : Ride |
	( r.reservation = re ) ) ) )
}

abstract sig Discount {}

one sig MoreThan2Passengers extends Discount {}

fact moreThan2PassengersCondition
{
	all ri:Ride, m2p: MoreThan2Passengers | m2p in ri.payment.discounts iff ri.passengers >=2
}

one sig EnoughBatteryLeft extends Discount {}

one sig CarPutInCharge extends Discount {}

sig Payment
{
	client: one Client,
	discounts : set Discount,
	appliedDiscount : lone Discount
}
{
	( appliedDiscount in discounts ) and
	( #discounts > 0 implies appliedDiscount != none )
}

fact paymentIsUnique
{
	( all disjoint ri1, ri2 : Ride | ri1.payment != ri2.payment or ( ri1.payment = none and
ri2.payment =
	none ) ) and
```

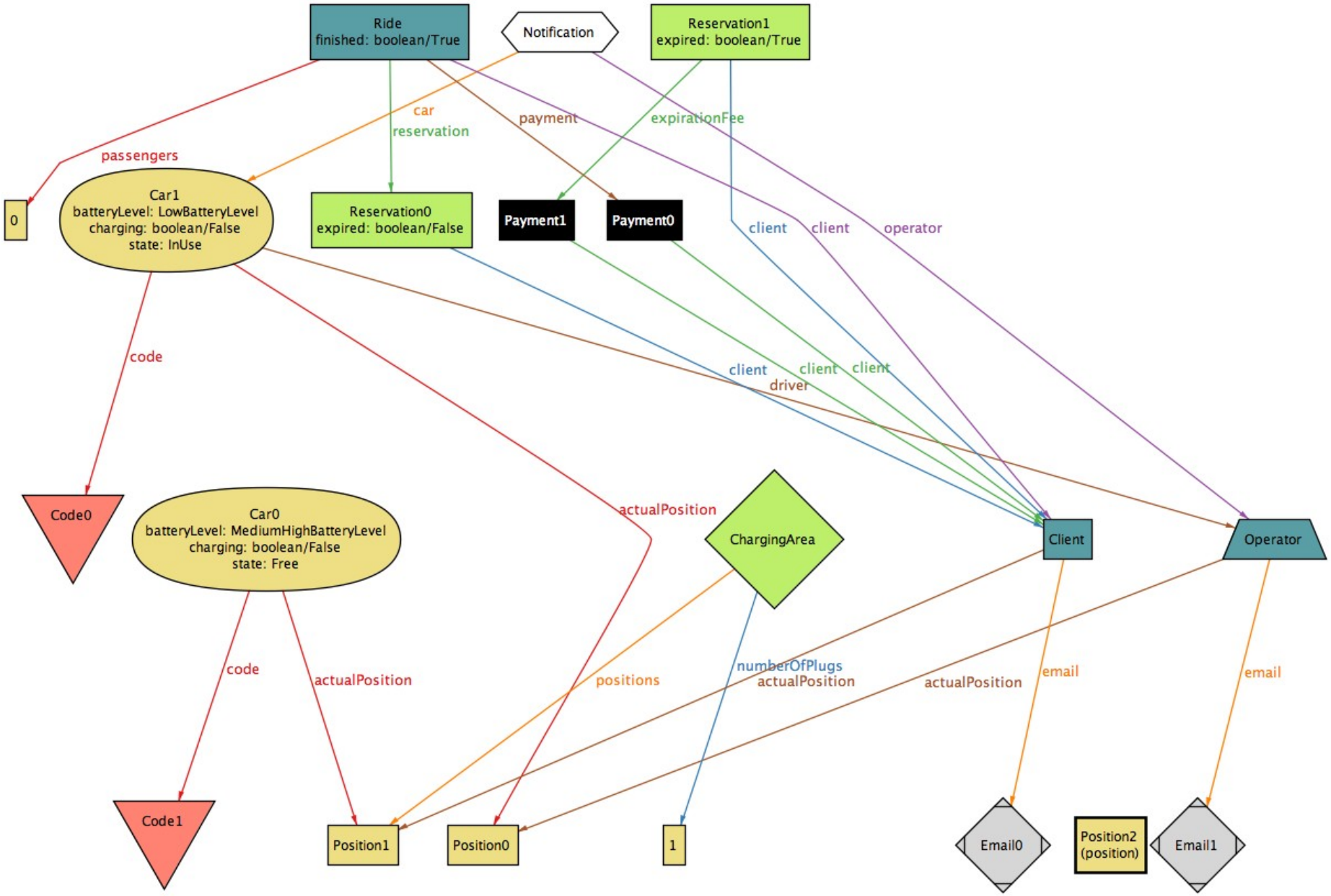
```
	( all disjoint re1, re2 : Reservation | re1.expirationFee != re2.expirationFee or
( re1.expirationFee
	= none and re2.expirationFee = none  ) ) and
	( all ri : Ride, re : Reservation | re.expirationFee != ri.payment or (re.expirationFee = none and
	ri.payment = none ) )
}

fact noStandalonePayments
{
	all p : Payment | ( one re : Reservation | re.expirationFee = p ) or ( one ri : Ride | ri.payment =
p )
}

fact positionOutSafeArea
{
	some position: Position | all sa: SafeArea | position not in sa.positions
}

fact clientThatReservesPay
{
	( all r : Ride |  r.finished = True implies r.client = r.payment.client  ) and
	( all re : Reservation | re.expired = True implies re.client = re.expirationFee.client )
}

fact onlyOneDiscountApplied
{
	all re:Reservation, ri:Ride, d,d1: Discount | (re.expired = True implies
	re.expirationFee.discounts = none) and
	(#ri.payment.discounts = 1 implies (ri.payment.appliedDiscount = d and d in
	ri.payment.discounts)) and
	(#ri.payment.discounts >1 implies
	(((d=CarPutInCharge and d in ri.payment.discounts) implies ri.payment.appliedDiscount = d)
	and ((d=CarPutInCharge and d1=EnoughBatteryLeft and d not in ri.payment.discounts and d1
in
	ri.payment.discounts)
	implies ri.payment.appliedDiscount = d1)))
}

fact discountOnlyOnRide
{
	all reservation : Reservation | ( reservation.expired = True ) implies (
	reservation.expirationFee.appliedDiscount = none and #(reservation.expirationFee.discounts)
=
	0 )
}

pred carIsInsideSafeArea [car : Car]
{
	one safeArea : SafeArea | InsideArea [car, safeArea]
}

pred carIsInUse [car : Car]
```

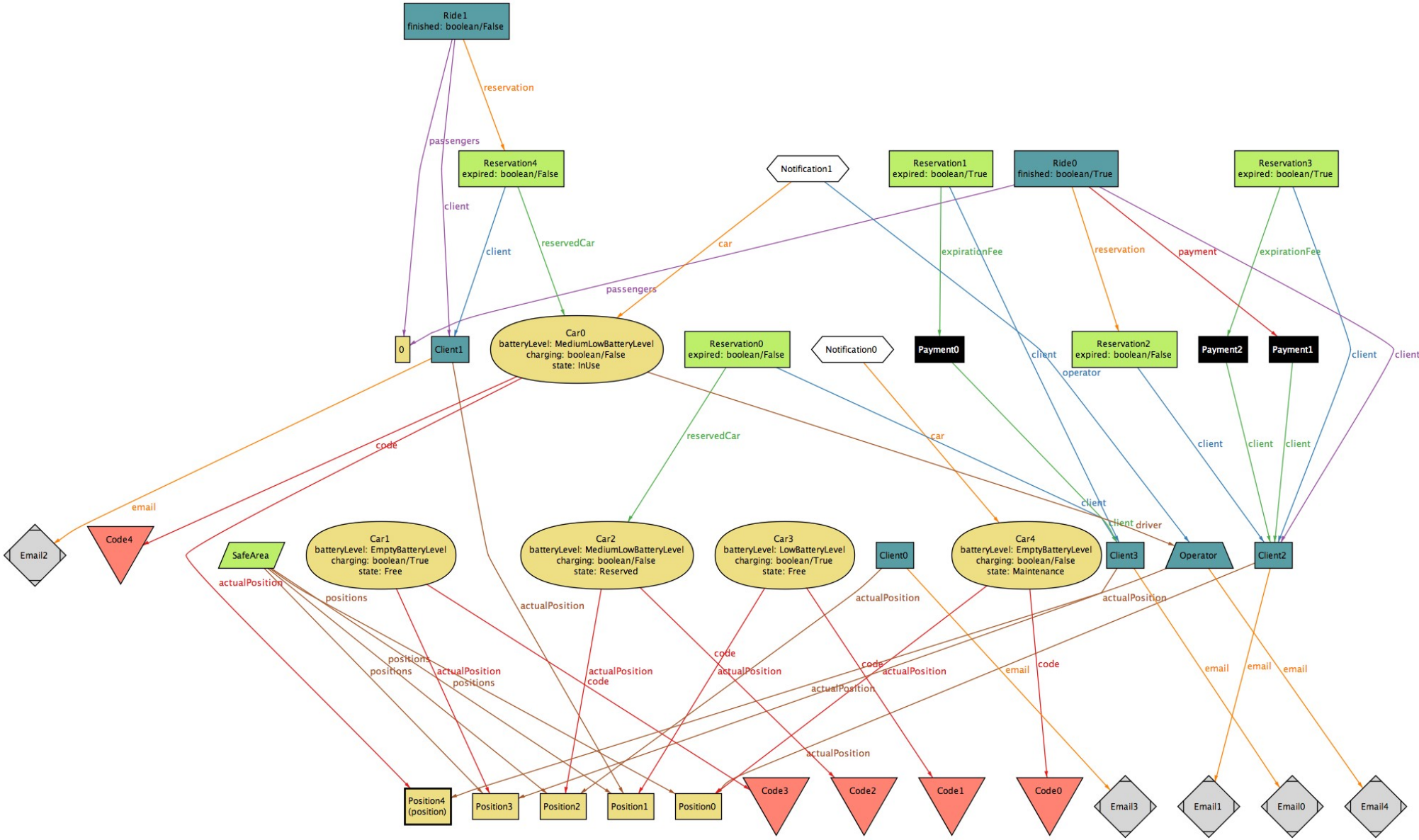
```
{
	one ride : Ride | ride.finished = False and ride.reservation.reservedCar = car
}

pred carNeedsMaintenance [ c : Car ]
{
	one notification : Notification | notification.car = c
}

pred notificationIsPending [ notification : Notification ]
{
	notification.operator = none
}

assert goalG4
{
	no disjoint reservation1, reservation2 : Reservation | reservation1.expired = False and
	reservation2.expired = False and reservation1.client = reservation2.client
}

assert goalG5
{
	all reservation : Reservation | reservation.expired = True implies reservation.reservedCar  =
none
}

assert goalG6
{
	all ride : Ride | ( ride.finished = False and ( no notification : Notification | notification.car =
	ride.reservation.reservedCar ) ) implies ( ride.client = ride.reservation.client and ride.client =
	ride.reservation.reservedCar.driver )
}

assert goalG7
{
	all ride : Ride | ride.finished = True implies ( some payment : Payment | payment.client =
ride.client )
}

assert goalG9
{
	all c: Car | ( c.state = Free implies ( carIsInsideSafeArea [ c ] ) ) and ( c.charging = True
implies ( (one ca: ChargingArea | c in ca.chargingCars) or c.state = Maintenance) )
}

assert goalG10
{
	( all reservation : Reservation | reservation.expired = True implies reservation.expirationFee !
=	none ) and
	( all ride : Ride | ride.finished = True implies ( ride.payment.appliedDiscount in
		ride.payment.discounts ) )
}
```

```
assert goalG11
{
        all c : Car | ( carNeedsMaintenance [ c ] ) implies ( ( ( one o : Operator , notification :
Notification      | notification.operator != none and notification.operator = o and notification.car = c ) or
( one
        notification : Notification | notificationIsPending [ notification ] ) ) )
}

pred show{
#Ride>1
#Notification>1
#Client>3
#Car>2}
run show for 8
check goalG4
check goalG5
check goalG6
check goalG7
check goalG9
check goalG10
check goalG11
```

4.2 Results




Ride
finished: boolean/True
Notification
Reservation1
expired: boolean/True
car
reservation
payment
expirationFee
passengers
0
Car1
batteryLevel: LowBatteryLevel
charging: boolean/False
state: InUse
Reservation0
expired: boolean/False
Payment1
Payment0
client
client
operator
code
client
client
client
driver
Code0
Car0
batteryLevel: MediumHighBatteryLevel
charging: boolean/False
state: Free
actualPosition
ChargingArea
Client
Operator
code
actualPosition
positions
numberOfPlugs
actualPosition
actualPosition
email
email
Code1
Position1
Position0
1
Email0
Position2
(position)
Email1


Ride1
finished: boolean/False
reservation
passengers
Reservation4
expired: boolean/False
client
client
reservedCar
Notification1
Reservation1
expired: boolean/True
Ride0
finished: boolean/True
Reservation3
expired: boolean/True
car
passengers
expirationFee
reservation
payment
expirationFee
0
Client1
Car0
batteryLevel: MediumLowBatteryLevel
charging: boolean/False
state: InUse
Reservation0
expired: boolean/False
Notification0
Payment0
client
Reservation2
expired: boolean/False
Payment2
Payment1
client
client
operator
code
reservedCar
car
client
client
client
client
client
email
Email2
Code4
SafeArea
Car1
batteryLevel: EmptyBatteryLevel
charging: boolean/True
state: Free
Car2
batteryLevel: MediumLowBatteryLevel
charging: boolean/False
state: Reserved
Car3
batteryLevel: LowBatteryLevel
charging: boolean/True
state: Free
Client0
Car4
batteryLevel: EmptyBatteryLevel
charging: boolean/False
state: Maintenance
Client3
client
driver
Operator
Client2
actualPosition
positions
actualPosition
actualPosition
actualPosition
positions
positions
positions
actualPosition
actualPosition
code
actualPosition
code
actualPosition
email
code
actualPosition
email
email
email
actualPosition
Position4
(position)
Position3
Position2
Position1
Position0
Code3
Code2
Code1
Code0
Email3
Email1
Email0
Email4

5 Used Tools

- Draw.io Diagrams: UML, Use Case, Sequence, State Chart diagrams
- Alloy Analyzer 4.2: model consistency checker
- Google Drive: documents sharing
- Google Docs: word processor, concurrent work platform
- GitHub: control version

6 Hours of work

Perugini Alex
- 24/10/16 1h 30m
- 26/10/16 2h
- 28/10/16 30m
- 04/11/16 1h 30m
- 05/11/16 5h 30m
- 06/11/16 4h
- 07/11/16 2h
- 08/11/16 2h
- 09/11/16 3h
- 10/11/16 2h
- 11/11/16 4h
- 12/11/16 3h 30m
- 13/11/16 6h

Re Marco
- 24/10/16 1h 30m
- 26/10/16 2h
- 28/10/16 1h
- 30/10/16 2h
- 31/10/16 1h 30m
- 01/11/16 2h
- 02/11/16 2h 30m
- 05/11/16 1h 30m
- 06/11/16 4h
- 07/11/16 2h
- 08/11/16 2h
- 09/11/16 3h
- 11/11/16 5h
- 12/11/16 5h 30m
- 13/11/16 6h

Scotti Vincenzo
- 24/10/16 1h 30m
- 26/10/16 2h
- 28/10/16 30m
- 30/10/16 1h 30m
- 31/10/16 1h
- 01/11/16 1h
- 02/11/16 2h 30m
- 04/11/16 1h
- 05/11/16 3h 30m
- 06/11/16 4h
- 07/11/16 2h
- 08/11/16 1h
- 09/11/16 3h
- 11/11/16 5h 30m
- 12/11/16 5h
- 13/11/16 6h

Changelog

- v 1.1
  - Rephrased the domain assumptions regarding the detections of the system
  - Added payment method and motorization actors in the use case diagrams
  - Added new situation to manage the breaking of a car:
    1. Assumption: System always detects correctly if a car is damaged or has a breakdown.
    2. Requirement in G11: The system must notify the nearest operator when a car is damaged or has a breakdown
    3. Use case: Repair damaged car (and relative table with description)
    4. Sequence diagram: Repair of a damaged car