POLITECNICO
MILANO 1863

Politecnico di Milano
2016-2017 Software Engineering 2 Project:
PowerEnJoy
Requirements Analysis and Specifications Document

Version 1.1

Alessandro Polenghi 879111

Alessandro Terragni 879112

Contents

1. Introduction

1.1. Purprose

We will project PowerEnJoy, which is a digital management system for a car-sharing service that exclusively employs electric cars.

1.2. Actual system

The company has already a system that handles all the data about the cars of the service, this system is called CMS.
With this system it's possible to detect:

- the position of the car
- the level of charge of the car
- if a car is out of order
- if an accident has occurred
- number of people present in every time in the car
- if a car is plugged in the charge station
- if the engine is working

1.3. Goals

PowerEnjoy should provide the functionality normally provided by car-sharing services.
These include:

• [G1] Users must be able to register to the system by providing their credentials and payment information

• [G2] Clients must be able to log-in in the service

• [G3] Clients must be able to find the locations of available cars in a prefixed range from their current location or from a specified address.

• [G4] Among the available cars in the range mentioned in the G3, user must be able to reserve a car

• [G5] The client must be able to pick up the car he has reserved

• [G6] The system must be able to charge the user for using the car

• [G7] = The system should be able to fine the client

• [G8] The system should be able to apply discounts to the clients

• [G9] The system should be able to make the client pay

• [G10] The system must be able to detect when an incident happened and when damages or failure occur

• [G11] The system must be able to allow the client to change its profile information

1.4. Glossary and Acronyms

To avoid misunderstanding and ambiguity, the following definitions and acronyms here listed represent the meaning that the developers associate to these common words in the context of the project:

- **Client** = a person who is logged in the system as a driver
- **User** = a person who is not registered in the system
- **GPS** = Global positioning System
- **Passengers** = all the people in a car, including the driver
- **SPA** = safe parking area, an area in which you can park without taking any fine.
  The set of safe parking areas is defined in the system.
- **RPA** = recharge parking area, an area in which you can park and recharge the car.
  The set of recharge parking areas is defined in the system.
- **CMS** = system that the company use to handle the data of the cars.
- **Active account** = status of a user account with verified and valid data
- **Suspended account** = temporary status associated to a client when his payment information is no longer valid
- **Valid payment information** = payment data that work properly (for example a no expired credit card)
- **Reservation timeout RT** = timeout of 1 hour, it starts when the car reservation is accepted and the OPC is sent.
  In this amount of time the client can open the reserved car with the OPC provided.
- **Pit stop timeout** = timeout of 5 minutes, it starts when the OPC code is entered in the car keyboard and the car unlocks.
  In this amount of time the client can start the engine of the car.
- **Put on charge timeout** = timeout of 5 minutes, it stars when a car stops in a RPA and the status changes from ‘riding’ to ‘to be charged’ or ‘available’ depending on the conditions.
  This timeout stops when the client plugs the car into the power grid.
- **PowerEnjoy opearators** = the company has a qualified core staff to repair and move cars whose status is ‘out of order’

- **Safety procedure** = procedure performed when an accident occurs or a damage/failure is detected by the car, this requires the participation of a PowerEnjoy opearator
- **Ride** = time in which the car stays in 'riding' status
- **OutOfSPAFine** = is a percentage that represent the fine to apply if a client parks the car out of an SPA.
  This fine not has not been defined by the customer, so it will be defined during the implementation
- **Parked car** = a car whose engine is turned off and the that is locked (all the car doors closed) without passengers inside; the parked car is not a status but a set of conditions.
  In fact, all the statuses are mutual exclusive, on the contrary the 'parked' conditions are verified in more than one status
- **Car status** = mutual exclusive condition of the car, it can be one of the following:
  - **Available car** = a car that is parked and available to rent because no one is using it and no one has already reserved it, moreover, it must have a level of charge > 20%
  - **Reserved car** = a car that has been booked by a client
  - **Out of order car** = a car that is broken or that had an accident
  - **To be charged car** = a car that is parked and whose battery level is <=20 %
  - **Riding car** = a car whose engine has been turned on

A more accurate description of the cars statuses and of the various types of fines and discount will be provided in the state diagrams of this document.

2. Overall Description

2.1. Product perspective

The application we will release is a mobile application which is integrated with one other existing system.
This application will not have any internal interface for administration but it will be only user based.

2.2. Client characteristics

The client that we expect to use our application is a person who wants an easy way to reserve and use a car from the car-sharing service.
All the cars are electric, so we can also think about a user that chose the PowerEnjoy service because is completely green.
Clients must own a mobile device with a GPS and internet data connection integrated, and have a valid credit card.

2.3. Constraints

2.3.1. Regulatory policies

The system must require to the client the permission to get his position and he has to manage sensible data (position, credentials and payment information) respecting the privacy law.
Furthermore, the systems must not use notifications to send SPAM respecting the privacy law.

2.3.2. Hardware limitations

Mobile app requires:

- 3G/4G connection
- GPS
- Space for app package

2.3.3. Interfaces to other applications

- Interface with e-mail gateway provider via standard e-mail APIs, to send emails to clients.
- Interface with the push service(s) via own APIs to send push notifications to clients.
- The PowerEnjoy service interfaces with the CMS, the actual system that the company use to manage its resources.

2.3.4. Parallel operation

PowerEnjoy must support parallel operations from different users at the same time.

2.3.5. Documents related

- Design Document (DD).
- Integration Test Plan Document (ITPD)
- Project Plan (PP)

2.4. Assumptions and Dependencies

- All the cars that a client can rent are already registered in the CMS system
- All the data that the CMS provides are correct
- Incidents, car failures or damages are already handled partially by the CMS: when an accident occurs or a car failure or a damage is detected, the CMS notifies the System that modifies the car status to 'out of order' and cancels the reservation and the payment.
  Afterwards, the reservation of the broken car is handled manually by a PowerEnjoy operator: the operator follows a standard procedure: he collects all the data about the current reservation, goes where the car is and will manage the reparation and all eventually the payment.
  This procedure is outside of the system that we will implement.
- After 'Out of order' cars are fixed by the operators, their status is changed manually to "available".
- Car maintenance is already handled by CMS and it take place periodically from the PowerEnjoy operators
- The communications between the system and the clients work correctly
- All the cars have a numerical keyboard in which the client insert the OPC to unlock the car and enter
- The system constantly interacts correctly with the CMS database to pool the data it requires
- The CMS database is updated constantly with all the cars information

- Changes in the car status like:

    - Failures
    - Switching on of the engine
    - OPC entered in the car keyboard
    - Opening and closing of the car doors

  trigger the CMS, that notifies the system.
- The values of the discounts and fines the client are fixed because they are required by the company
- Payments operations always go well

2.5. Actors identifying

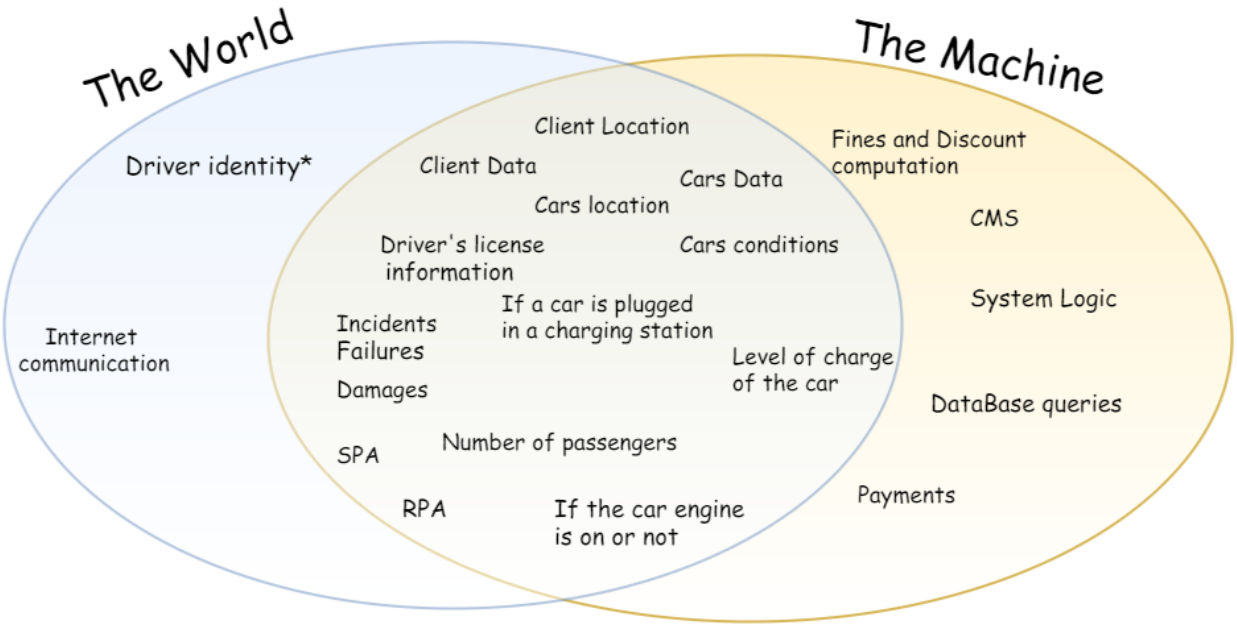
The actors of our system are:

- User = a person who is not registered yet in PowerEnjoy
- Client = a person who has registered to the PowerEnjoy and is logged in the system

3. Requirements

3.1. The world and the machine

For a first domain analysis of the PowerEnjoy system we use “The World & Macchine” model by M. Jackson & P. Zave.



***It’s not possible to detect the real driver of the car: the OPC code can be entered by anyone**

3.2. Functional requirements

Assuming that the domain properties described in the paragraph 2.4 hold, in order to completely satisfy the goals defined in the paragraph 1.3, the following requirements can be derived.

- **G1: user registration**
  - The system must be able to check if the email chosen by the user is valid and doesn't belong to other clients
  - The system must be able to check if the payment information is valid
  - The system must be able to check if the driving license data are correct
  - If the previous three controls go well, the system must save the user information and payment information
  - The system must be able to generate a password and sends it to the client email provided

- **G2: client log-in**
  - The system must be able to check if the email and related password provided are correct
  - The system must only let the client in if the provided username and related password are correct
  - The user must have already done the registration process

- **G3: search available cars**
  - The system must be able to find the client position
  - The client must be logged in
  - The system must be able to show to the client the cars available in a range of fixed number of km
  - The system must be able to interact with the CMS database to retrieve the available cars information

- **G4: reserve a car**
  - The client must be logged in
  - The system must be able to check if the account is active
  - The client can't reserve a car if the account is suspended
  - The client can't reserve more than one car at the same time

- The client must be able to choose a car from the ones available
- The system must be able to check if the car chosen by the client is still available
- The system must be able to change the status of the chosen available car to ‘reserved’
- The reservation must be active for a fixed amount of time
- Once the car is reserved, the system must be able to send an OPC to the client connected with the car reserved
- The client can reserve a car only if the payment information is not expired
- When the payment information expires, the system must suspend the client account until he uploads new valid payment information

- **G5: pick up the car**
  - The client can enter the car only if is still available
  - The client must enter in the car keyboard the OPC code that he has received during the reservation
  - The system must be able to check if the OPC is correct
  - If the OPC inserted is correct, the system should be able to open the car
  - The reserved car can be opened with the OPC only if the reservation timeout is not expired
  - As soon as it opens the car, the system should be able to stop the reservation timeout and start the pit stop timeout
  - The engine can be turned on only before the pit stop timeout haven’t expired
  - The system should be able to stop the pit stop timeout if the engine is turned on

- **G6: charging the ride**
  - Car must be in riding status
  - The system should be able to charge the client per each minute the car status is ‘riding’
  - The system must change the status of the car in ‘available’ only if the ‘parked’ conditions are satisfied, there are no damages detected and the battery level of charge is >= 20%

- The system must change the status of the car in ‘to be charged’ only if the ‘parked’ conditions are satisfied, there are no damages detected and the battery level of charge is < 20%

- **G7: to fine the client**

  - **fine for reservation timeout**
    - The system should be able to fine the client if the reservation timeout is expired before the OPC is entered in the car keyboard
    - If the reservation timeout expires, the system disables the OPC code previously provided.
    - If the reservation timeout expires, the system sets the car available again to be reserved: its status became ‘available’ again

  - **fine for pit stop timeout**
    - The system should be able to fine the client if the pit stop timeout is expired before the engine is turned on
    - The system should be able to fine the client for each minute he stays in the car after the pit stop timeout expires
    - The system should be able to check if the ‘parked’ conditions hold
    - If the ‘parked’ conditions hold, the system should be able to change the status the car to ‘available’

  - **parking fine**
    - The system should be able to detect the position of the parked car
    - The system should be able to fine the client if the car is not parked in a SPA or a RPA

  - **bad charge battery status fine**
    - The ‘parked’ conditions of the car must hold

- The system must be able to check if the car is parked in a safe area
- The system must be able to check the battery status of the car
- If the status of battery is <20%, the system records the bad charge battery status fine

- **Parking distance fine**
  - The 'parked' conditions of the car must hold
  - The system must be able to check if the car is parked in a SPA
  - The system must be able to check how distant is the nearest RPA
  - The system must be able to check if the nearest RPA is below the distance of 3Km
  - If the nearest RPA is above the distance of 3Km, the system records the fine for the parking in a distant position

- **G8: Giving discounts**

  - **car sharing discount**
    - The engine must be turned on
    - The system must be able to check how many peoples there are in the car during the ride
    - The system records the maximum number of passengers during the ride
    - If the number of passenger is >2 the system records the discount

  - **RPA discount**
    - The client must have already parked the car in a RPA and the 'parked' conditions must hold.
    - The system must be able to check if the car is parked in a RPA
    - When the car status 'riding' change in 'to be charged' or 'available' (the car hold the 'parked' conditions) the "put on charge timeout" starts

- The "put on charge timeout" stops only if the car is plugged to the power grid
- If the car is plugged before the "put on charge timeout" expires, the system records the discount

- **Good charge battery status discount**
  - The car must hold the 'parked' conditions
  - The system must be able to check if the car is parked in a SPA
  - The system must be able to check how distant is the nearest RPA
  - The system must be able to check if the nearest RPA is below 3Km of distance
  - The system must be able to check the battery status of the car
  - If the status of battery is >= 20% and the nearest RPA is below the distance of 3Km, the system records the good charge battery status discount

- **G9: payment**
  - Car status must have been changed from 'riding' to 'to be charged' or 'available' depending on the conditions
  - The system must be able to calculate the right amount of money due with the fines and discount recorded
  - The system must be able to executes the bank transaction
  - The system must be able to send the payment receipt with all the details

- **G10: incident, damages and failures detection**
  - If, during the ride, an incident occurs or any damages or failure are detected by the car, the status must able to change in 'out of order'
  - The system must able to show an error message on the car screen and invite the client to stop the car
  - The system must be able to alert the PowerEnjoy operator about the problem

- The system must be able to transfer all the data about the reservation to the system managed by the operator
- The system must be able to cancel the reservation and the payment
- The system should be able to notify the client about the cancellation

- **G11: Managing client profile**
  - The system should be able to let the user change its password
  - The system should be able to let the user change its email
  - The system must allow the user to change the email only if the new email is valid and doesn't belong to other clients
  - The system should be able to let the user change its payment information
  - The system must allow the user to change the payment information only if the new payment information is valid

3.3. Non Functional Requirements

Some non-functional requirements have been found in order to fulfil some qualities and a correct operation flow that the system should provide:

- **User friendliness**. The use interface has to be as simple and intuitive as possible. This because we want to simplify as much as possible the process to encourage the car sharing phenomenon. Furthermore, we want to make a system that could be accessible from much of the urban population.

- **Portability**. The client has to be compatible to all the major hardware and software platform on the market, this is accomplished using an application that fits well all the most popular mobile operating systems: Android and iOs.

- **Performance**. To supply suitable service, the system has to be reactive and able to answer to a high number of simultaneous requests. That's why the interactions between the client and the server have to be reduced as much as possible, in order to not overload the net.

- **Reliability**. The system should be able to guarantee the service 24/24, 7/7. The server farm should allow every day maintenance without compromise the functionality. However, the system doesn't cover a critical function, so brief unavailability could be acceptable.

- Data **integrity**, **consistency** and **availability**. Data have to be always accessible, so the system has to provide always an access to them in normal condition.

- **Security**. To ensure that private sensible data such as personal information and data bank remain private, the system should offer different security measures. First of all, all the account will be protected by the couple personal e-mail plus personal password. Secondly, since the system allows the insertion of data, such data should be filtered in order to avoid vicious or unwanted modification in the data base (e.g. SQL injection).

4. Scenarios

4.1. Scenario 1

Walter White, chemistry teacher, always takes the subway to go to his office but, unfortunately, today there is a strike, so he is forced to find another way to go there.
From a public advertising, he has heard about PowerEnjoy, an electric car sharing service.
Thus, he downloads the app on its smartphone, opens the app, and fills in the registration form with its data.
When he completes it, he receives the confirmation mail with the password, and he logs-in in the app.
Then, his wife Skyler arrives home and offers him a lift, so he closes the app, complaining about the time wasted.

4.2. Scenario 2

Hank, a police officer, is a usual costumer of the PowerEnjoy app, and needs a car to go to the city hall.
After logging-in in the app, he looks for the nearest cars from its position, then, he chooses one from the list of cars available and ask for a reservation.
The app confirms its reservation and notifies him with the OPC code.
Hank walks to the chosen car in 20 minutes, it opens the car entering the OPC code in the car keyboard, and immediately starts the engine.
He drives alone to its destination and it leaves the car in a SPA with a battery level of 33%.
Once it has closed the car, it receives a notification with the bill for the ride without any discount applied.

4.3. Scenario 3

After some weeks from its registration to PowerEnjoy, Walter decides to try again the service.
So, after logging-in in the app, he manages to reserve a car.
After discovering that he has an hour to reach the car, he decides to take a nap but unfortunately he fells deeply asleep.

He wakes up after 60 minutes with the sound of a notification on its smartphone: he has received a fine of 1€ because it has reserved a car but he hasn't reached it in time.

4.4. Scenario 4

Thinking that the third time should be the good one, Walter tries again to use the PowerEnjoy app.
He logs in and he reserves a car, then he goes immediately to it and he enters the OPC code, managing to unlock the car.
Then he enters the car, when he suddenly sees a countdown of 5 minutes starting.
In that moment, he hears the phone ringing and he answers to the call.
After 6 minutes of arguing with his colleague Jesse, he is ready to start the engine.
He tries to turn on the car but without any success...
Then its phone rings again: he has received a notification that addresses him a fine because he hasn't turned on the engine in time.
Very disappointed, he leaves the car closing the car door, forgetting the smartphone inside.
He tries to open the car again but it has closed automatically, also the code that he has let him enter at the beginning doesn't work anymore.

4.5. Scenario 5

Hank is driving in the PowerEnjoy car as usual when suddenly he starts feeling very bad.
He needs to stops as soon as possible but unluckily, there isn't any SPA or RPA nearby.
So he decides to park the car in a normal parking slot and he leaves the car there, looking for help.
As he knows, he receives a notification informing him about the fine of 10 € for having parked the car outside the predefined spots.

4.6. Scenario 6

Jesse is a user of the PowerEnjoy app and he has reserved and taken a car for him and his girl to go in the city-center and meet some friends.
His girlfriend Jane, disagrees about the choice of taking a car because

she expects that there will be traffic in the center of the city at this hour of the day.
Jesse don't want to listen to Jane since she proposes to take some bikes and he don't want to ride and toil.
Only after 5 minutes, they enter in the middle of rush hour traffic, so Jane start arguing saying she was right and continues claiming she is never considered when she says something.
Jesse, distracted by the discussion, bumps violently the car next to him, breaking all the hood.
The car is now useless and Jane is anger.
The CMS detects immediately the incident, it notifies the PowerEnjoy that starts the security procedures.
Jessie receives a notification about the cancellation of the reservation that informs him also about the coming of a PowerEnjoy operator

4.7. Scenario 7

After his bad experiences with the PowerEnjoy service, Walter wants to try to use it again to go shopping at the shopping center with his wife Skyler and his son Walter Jr.
Today he has reserved and taken the car with his family and after 15 minute they finally arrive at the shopping center.
Walter, to save money, has chosen a center where there are a lot of RPA. In fact, he had read that when someone park in a RPA and plug the car into the power grid, the system applies a discount of 30% on the last ride. Moreover, he knows about the 10% car sharing discount.
Walter today feels very cunning since he has planned to make his family happy saving a lot of money.
As soon they parked the car, Walter takes care of plugging the car into the power grid and eager wait the payment notification. The Walter's smartphone rings in Skyler's bag, the notification has arrived, but Skyler suspiciously checks the message and finds out that they have saved the 40% on the last ride.
Skyler exults announcing that she can afford the bag she wanted, making Walter very upset.

4.8 Scenario 8

Saul is a prominent lawyer. For some day his Cadillac has been broken, so he has registered to PowerEnjoy to reach its customers in all the part of his city.
This morning he received a call about an appointment in the afternoon, so he has reserved and taken a car and he has driven to the address of his costumer.
When he arrives, he parks the car in one of the SPA and checks that the level of charge is above the 20% to avoid the fine.
After a few minute his smartphone rings, he has received the notification of payment. On the retail payment he sees the entry “fine for the parking distance” with an extra 30% charge on his ride.
In that moment he realizes that he has parked at least 3 Km far from the nearest power grid station. He is now very angry with himself for having accepted that call.

4.9. Scenario 9

Hank is driving on the PowerEnjoy car to go to Los Pollos Hermanos, a popular fast food nearby.
When he arrives in the parking, he sees that this selling point is closed. He doesn’t want to give up his usual lunch, so he drives for 45 minutes to the other selling point of the city.
Once arrived, he parks in a SPA and hungry runs to the restaurant.
Some moments later he receives the payment notification, he checks it and sees that the system had charged 30% more on the last ride to compensate for the cost required to re-charge the car, in fact, the car has the battery charge level = 15%.
By the way, it doesn’t really care right now because his lunch is waiting for him to be eaten.

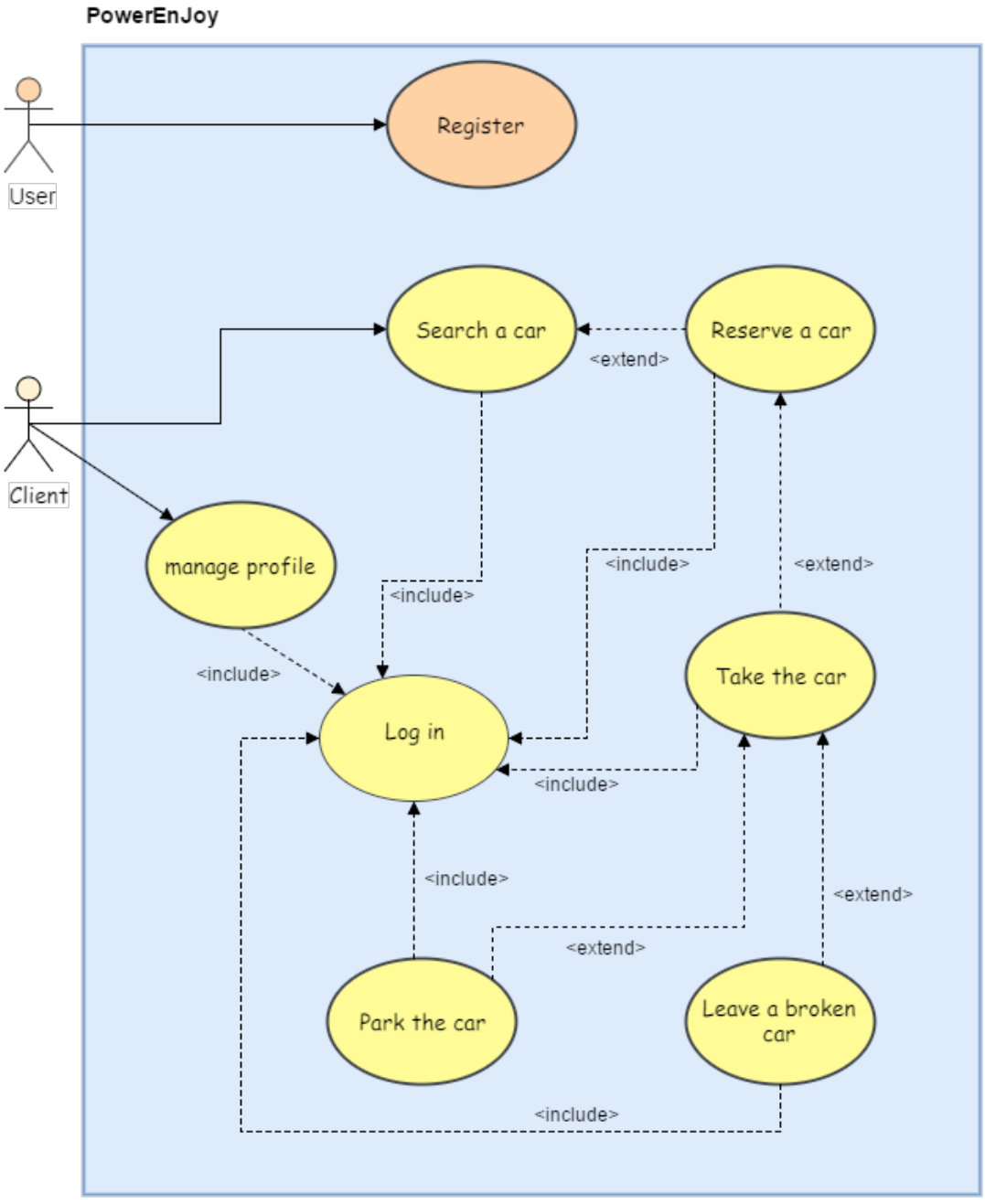
4.10. Scenario 10

Marie, Hank’s wife, usually uses the PowerEnjoy service, since she is a fierce environmentalist and wants to use only electrical cars.
This evening she had reserved and taken a car to go to her sister.
Skyler home is quite close to her house and after driving for 10 minutes she arrives.

She parked the car in one of the SPA with a battery level of 80%. While she is ringing the bell, her smartphone rings: the payment notification is arrived.
Reading it, she notices the she had saved the 20% on the last ride because the battery level was over the 50% of charge.
Feeling happy, she finally enters the house.

5. UML models

5.1. Use case diagram



5.2. Use cases

5.2.1. Use case 1

Name: Register

Actors: user

Entry conditions:

- User fills all the required fields in the registration page
- User clicks the button ‘register’

Flow of events

- PowerEnjoy checks if all the email provided by the user belongs to other existing clients
- PowerEnjoy checks if the driving license id is valid
- PowerEnjoy validates the payment information
- PowerEnjoy saves the user profile
- PowerEnjoy generates a password connected to the user profile
- PowerEnjoy sends the password to the email provided by the user

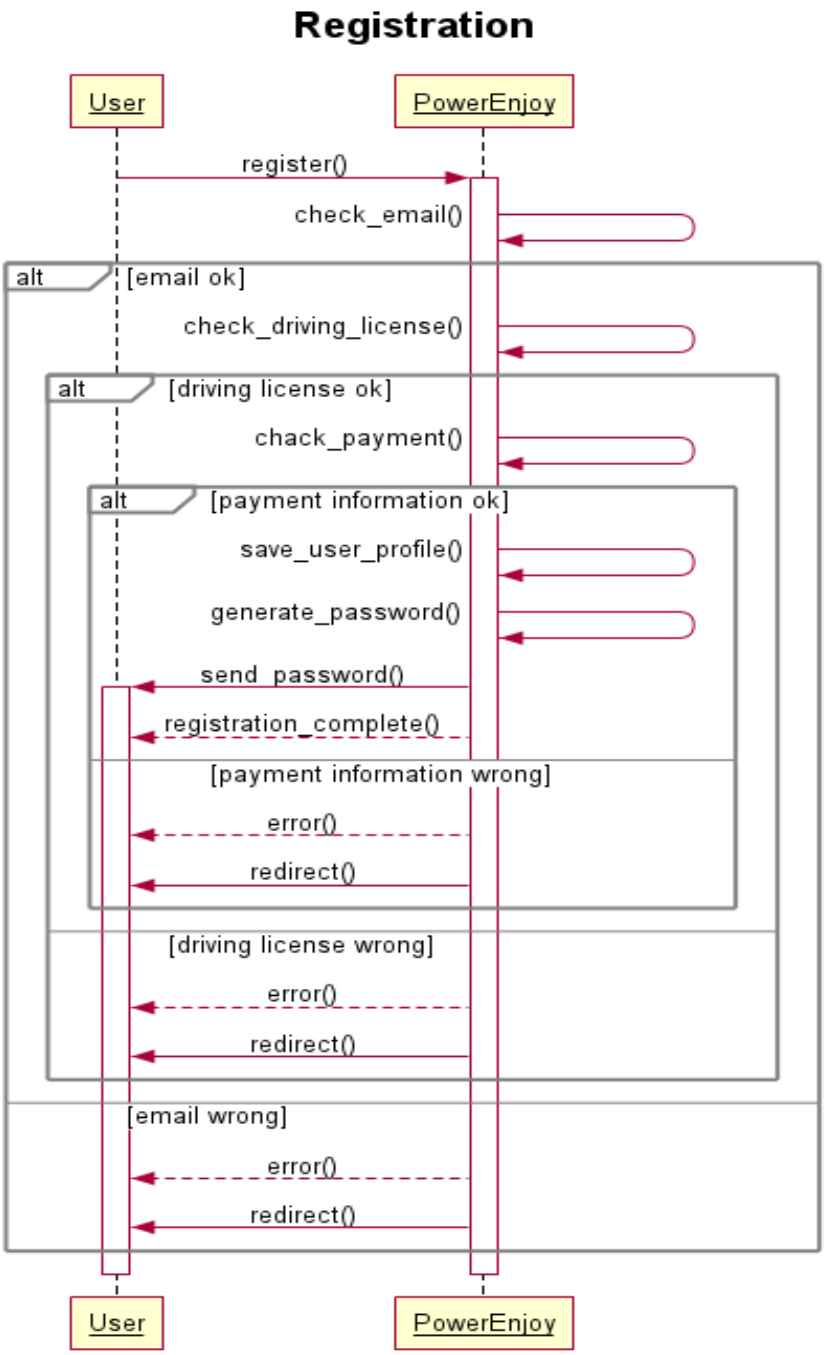
Exit condition

- PowerEnjoy notifies the user with a ‘registration complete’ message

Exceptions

- The email provide by the user already belongs to an existing client, in this case PowerEnjoy, doesn’t send the password and doesn’t notify him with the ‘registration complete’ message but notifies the user with an error and redirects him to the registration page, allowing him to refill the email field
- The payment data are not valid, in this case PowerEnjoy notifies the user with an error and redirects him to the registration page, allowing him to refill the payments fields
- The driving license id is not valid, in this case PowerEnjoy notifies the user with an error and redirects him to the registration page, allowing him to refill the driving license field

Sequence diagram



5.2.2. Use case 2

Name: Login

Actors: Client

Entry conditions:

- The client must be already registered
- The client must be in the login page of the PowerEnjoy app

Flow of events:

- The client fills the login fields with his email and password provided during the registration
- The client clicks on the login button, sending a login request to PowerEnjoy
- PowerEnjoy checks if the client's email and password are correct

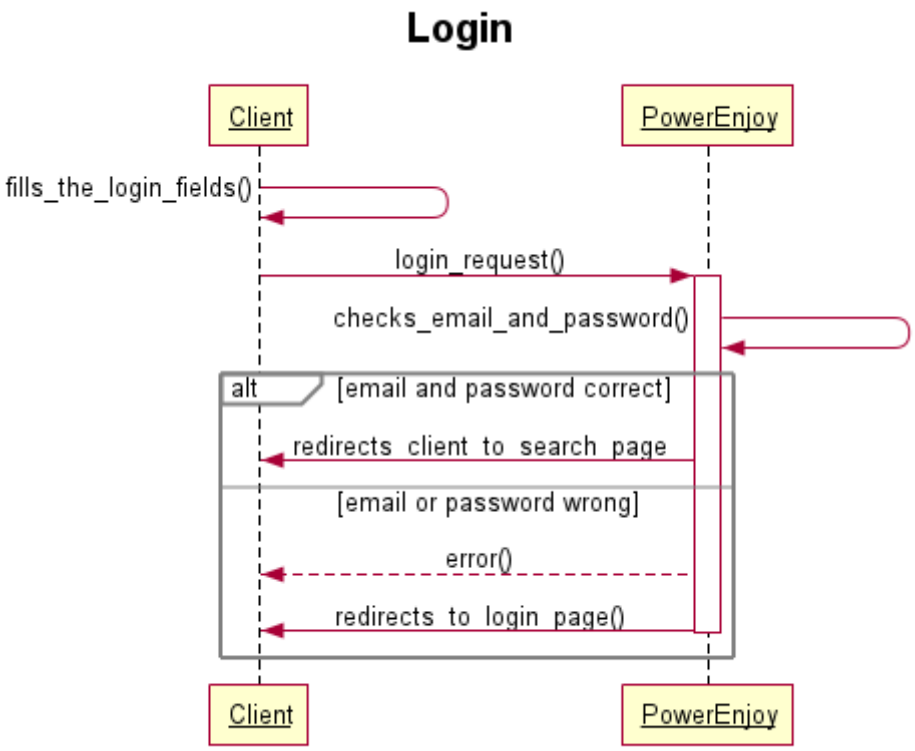
Exit conditions:

- PowerEnjoy redirects the client to the search page of the app

Exceptions:

- If the email or password entered by the client are wrong, PowerEnjoy doesn't redirect the client to the search page but notifies him with an error.
  Then, it allows the client to fill the login fields again.

Sequence diagram



5.2.3. Use case 3

Name: Search a car

Actors: Client

Entry conditions:

- Client must be already logged-in in the PowerEnjoy app
- Client must be in the search page of the PowerEnjoy app
- Client selects 'search from my position' button or inserts a specified address to start the search

Flow of events:

- If 'search from my position' button has been selected, PowerEnjoy detects the position of the user
- If the user has inserted the address, PowerEnjoy detects the position of the address
- PowerEnjoy retrieves from the CMS the cars available in a fixed range from the position of the user or from the address entered

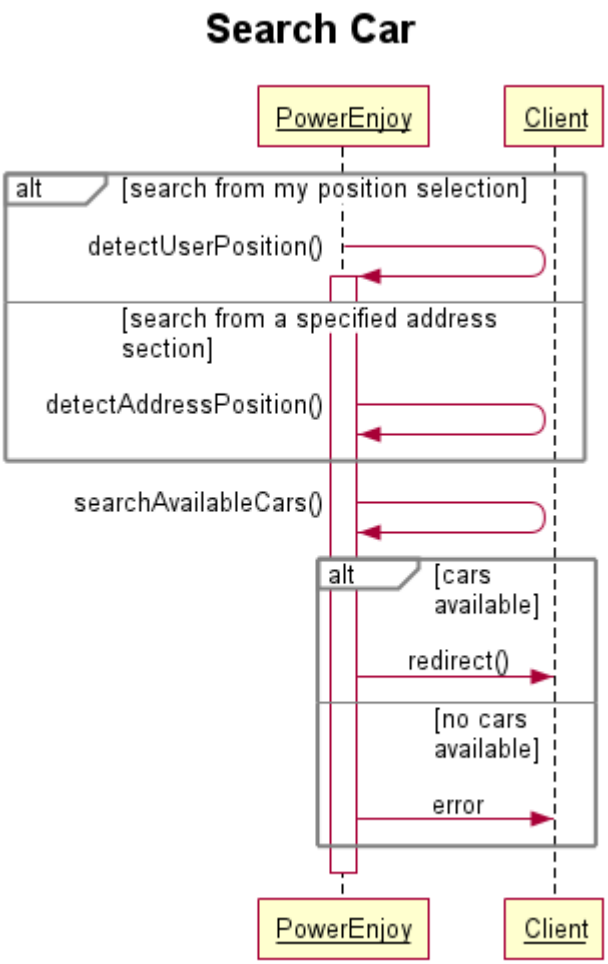
Exit conditions:

- PowerEnjoy redirects the client to the 'select car' page

Exceptions:

- If there are no cars available, PowerEnjoy doesn't redirect the client to the 'select car' page but it notifies him with an error and invites him to try to insert another address or to retry later

Sequence diagram



5.2.4. Use case 4

Name: Reserve a car

Actors: Client

Entry conditions:

- The client must be in the 'select car' page
- The client mustn't have already a reserved car or riding car associated with his profile at the moment of booking
- The client can't reserve a car if the account is suspended

Flow of events:

- The client selects one of the car in the 'select car' page
- The client sends a reservation request for that car to the PowerEnjoy
- If the payment information associated to the client are not expired the reservation keeps on
- If the car is still available, PowerEnjoy accepts the request
- PowerEnjoy notifies the client about the successful reservation
- PowerEnjoy generates an OPC connected to the car reserved
- PowerEnjoy changes the status of the car from 'available' to 'reserved'
- PowerEnjoy starts the reservation timeout

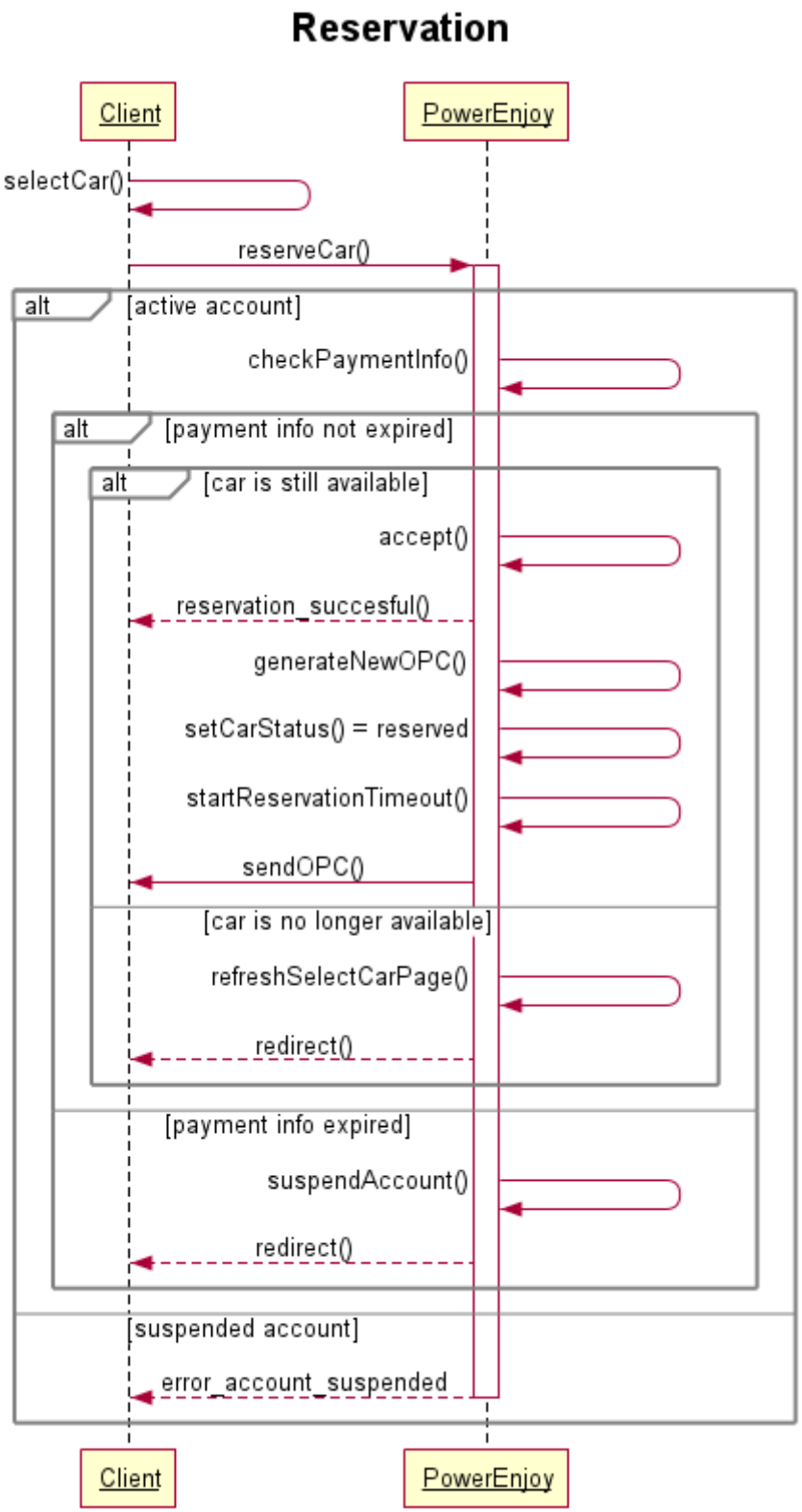
Exit conditions:

- PowerEnjoy notifies the client with an OPC to open the chosen car

Exceptions:

- If the client account is suspended, PowerEnjoy show a message of error which invite the client to update the payment data
- If the payment information associated to the client are expired, the PowerEnjoy suspends the client account and ends the reservation
- If the car is no available, PowerEnjoy rejects the reservation request and the clients is redirected to the updated 'select car' page

Sequence diagram



5.2.5. Use case 5

Name: Take a car

Actors: Client

Entry conditions:

- Client must have already reserved a car
- Client must have already received the OPC code
- Client must be already in front of the car

Flow of events

- The client inserts the OPC code into the car keyboard
- PowerEnjoy checks if the OPC code is correct AND the reservation timeout is expired AND the car status is not 'out of order'
- If the OPC code entered is correct and the reservation timeout is not expired and the car is not out of order, PowerEnjoy opens remotely the car
- When the car is opened, PowerEnjoy starts the pit Stop timeout
- The client and his possible passengers enter the car and they get ready to leave
- The client tries to starts the engine
- PowerEnjoy checks if the pit Stop timeout is not expired
- If the pit Stop timeout is not expired the PowerEnjoy allows the engine to start

Exit condition

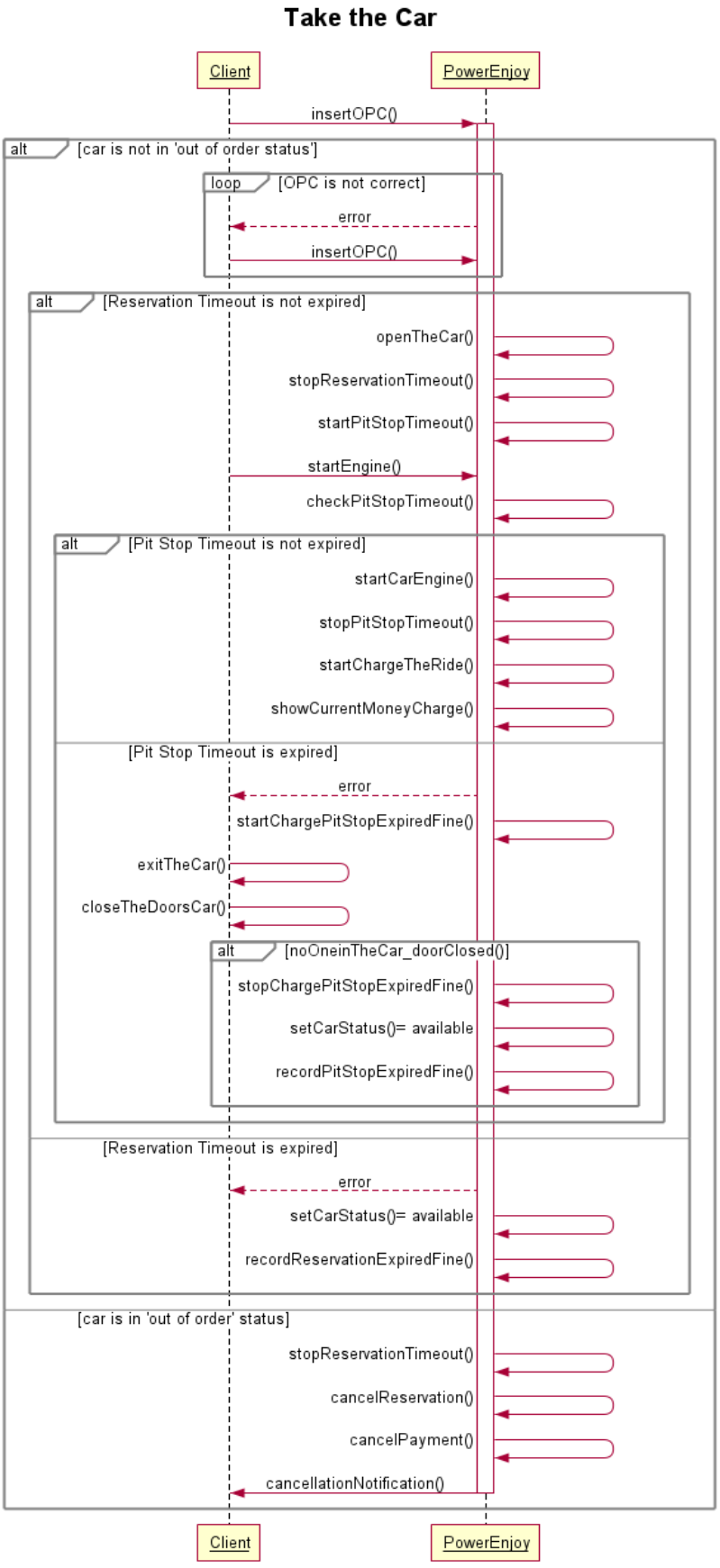
- PowerEnjoy starts to charge the client per each minute from the starting of the engine
- PowerEnjoy shows on the car screen the current amount of money charge the client

Exceptions

- If the reservation timeout is expired, PowerEnjoy doesn't allow to open the car.
  PowerEnjoy charges the client with a fine.
  PowerEnjoy sets the car available again to be reserved: its status became "available" again

- If the pit Stop timeout is expired, PowerEnjoy charges the client with a fine calculated for each minute he/she and his possible passenger stays into the car.
  PowerEnjoy stops to charge the fine as soon as the parked conditions hold, PowerEnjoy sets the car available again to be reserved: its status became “available” again.
- if the car is out of order, it can’t be opened with the OPC code and the reservation timeout is stopped.
  In this case, PowerEnjoy cancels the reservation and the payment.

Sequence diagram



5.2.6. Use case 6

Name: Park the car

Actors: Client

Entry conditions:

- The client turns off the engine and exits the car with all the passengers

Flow of events

- PowerEnjoy checks the 'parked' conditions
- If the 'parked' conditions are satisfied, PowerEnjoy stops the charging of money
- PowerEnjoy checks if the car is not out of order
- PowerEnjoy checks the position of the car
- PowerEnjoy checks the battery charge level
- PowerEnjoy checks the distance to the nearest RPA
- PowerEnjoy checks the maximum number of people present into the car during the ride
- If the car is left out of a SPA or RPA, PowerEnjoy records the fine for the parking in position not allowed: outOfSPAFine, in this case no other fine or discounts are addressed
- If the car is parked in a RPA PowerEnjoy starts the "put on charge timeout"
- When the "put on charge timeout" stops, PowerEnjoy records the related discount of 30%
- PowerEnjoy calulates the nearest RPA from the parked car position
- If the car is parked in a SPA, the status of battery is >= 20% and the nearest RPA is below the distance of 3Km, PowerEnjoy records the good charge battery status discount: 20%
- If the car is parked in a SPA and the level of battery is <20%, PowerEnjoy records the bad charge battery status fine: 30%
- If the car is parked in a SPA and the nearest RPA is above the distance of 3Km, PowerEnjoy records the fine for parking in a distant position: 30%
- If the number of passenger is > 2, PowerEnjoy records the discount: 10%
- PowerEnjoy calculates the final price
- PowerEnjoy performs the payment

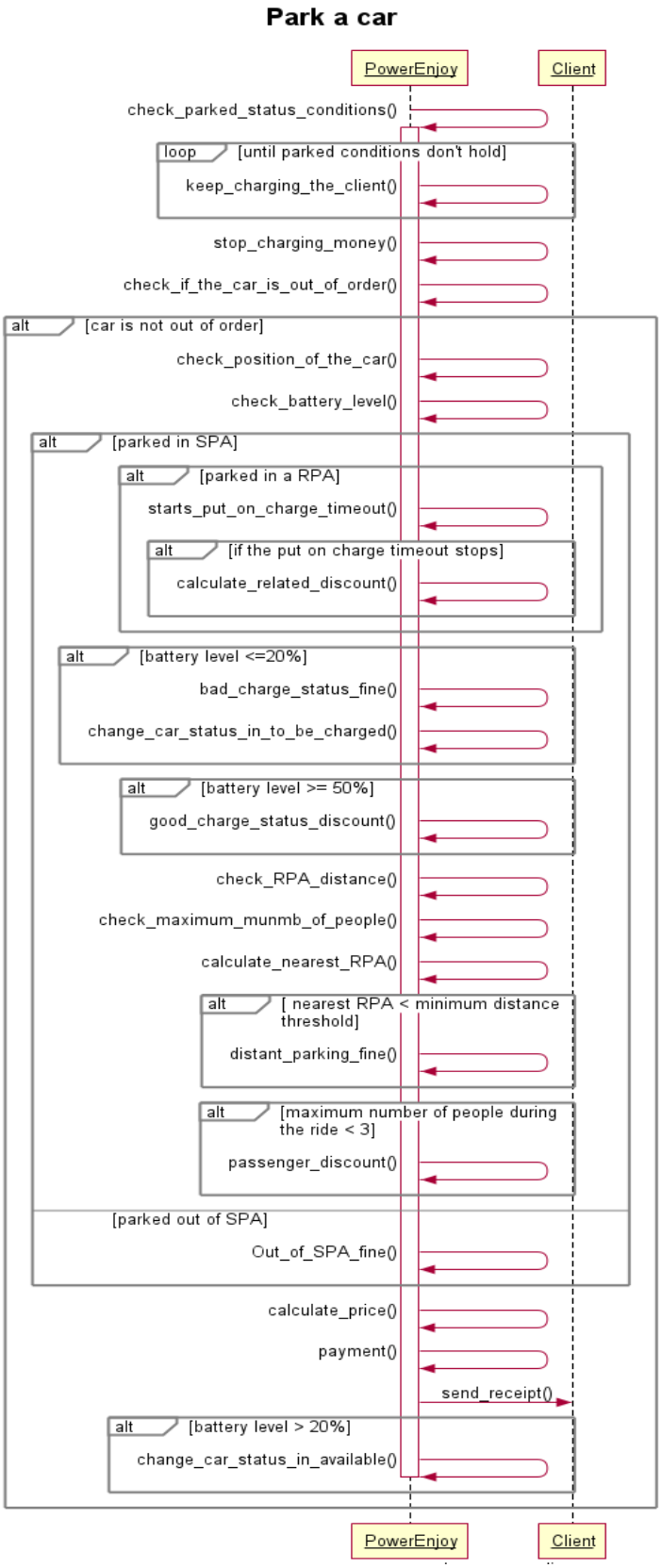
Exit condition

- PowerEnjoy sends the receipt to the client
- If the level of charge is > 20% AND no failure or damages are detected the car status is changed to ‘available’

Exceptions

- If “put on charge timeout” expires, PowerEnjoy doesn’t address the related discount to the client
- If the status of battery is < 20% and is not in ‘out of order’ status, the car status becomes ‘to be charged’

Sequence diagram



5.2.7. Use case 7

Name: Leave a broken car

Actors: Client

Entry conditions:

- PowerEnjoy detects an incident or a failure or a damage of the car when the car status is ‘riding’

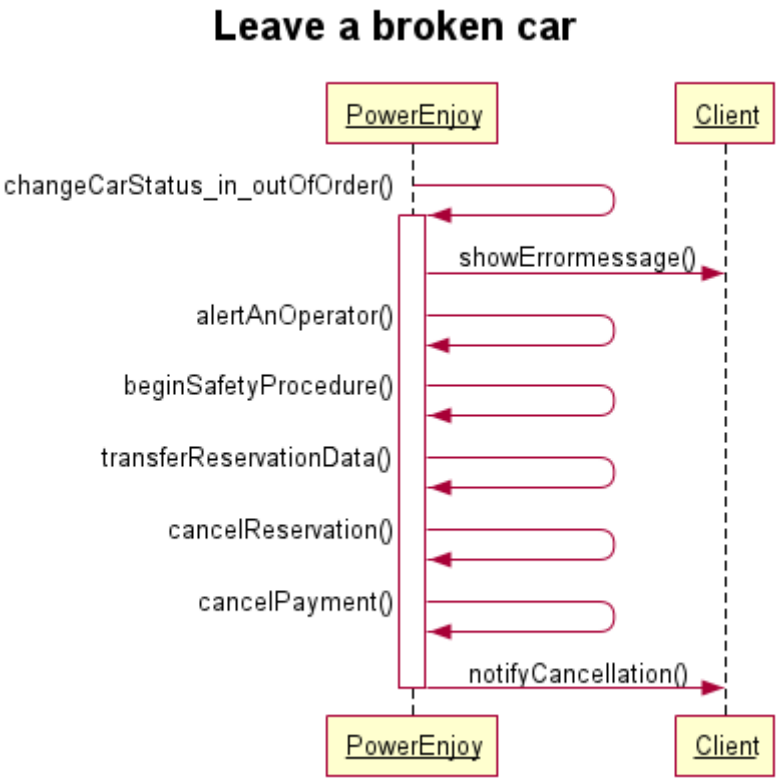
Flow of events

- PowerEnjoy change the car status in ‘out of order’
- PowerEnjoy shows, on the car screen, a message which invite the driver to stop the car
- The client must stop the car because the damage doesn’t allow to go on safely
- PowerEnjoy alerts an operator

Exit condition

- PowerEnjoy begin the safety procedure
- PowerEnjoy transfer the reservation data to the operator system
- PowerEnjoy cancels the reservation and the payment

Sequence diagram



5.2.8 Use case 8

Name: Managing profile

Actors: Client

Entry conditions:

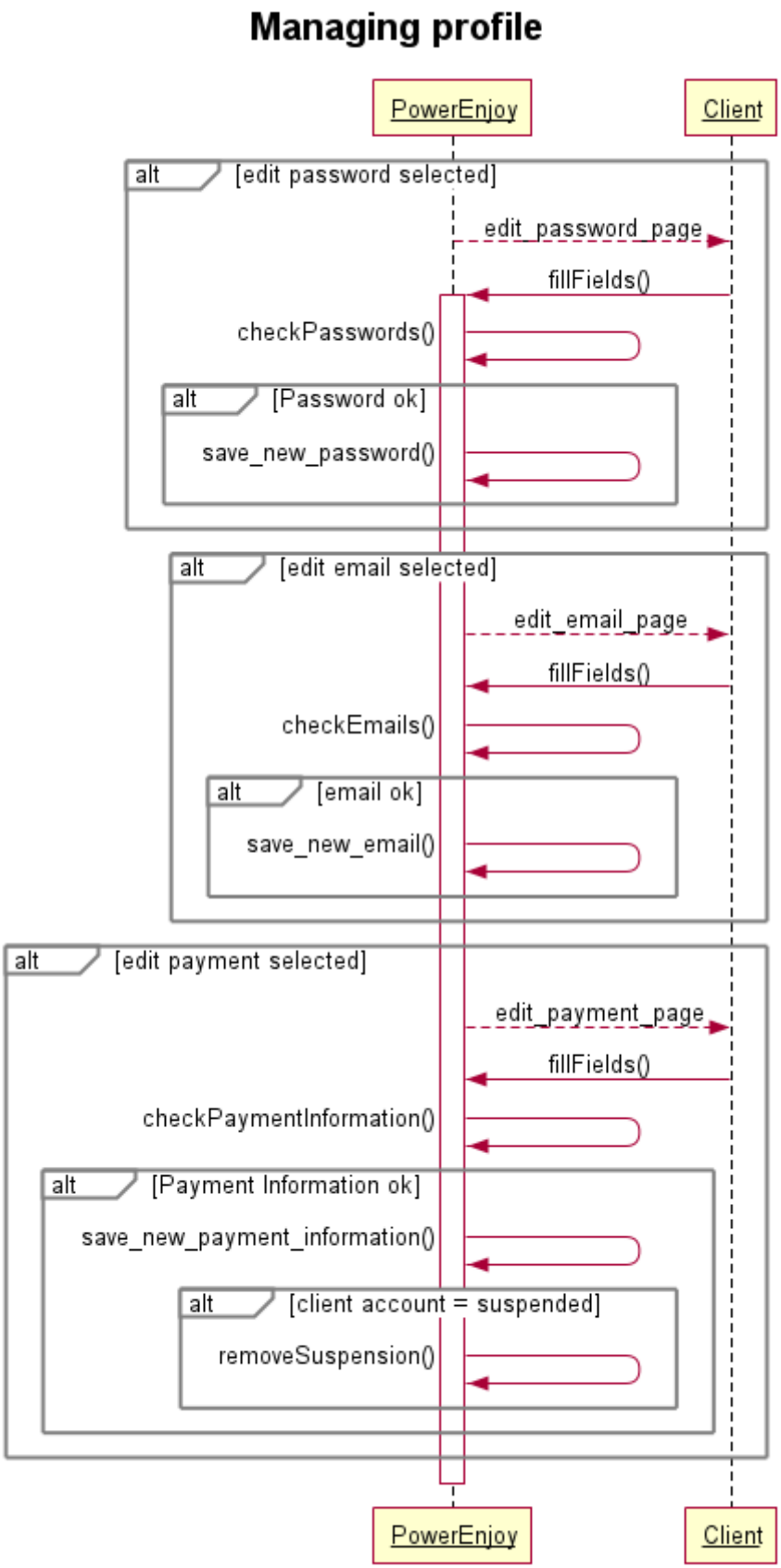
- Client must be already logged-in in the PowerEnjoy app
- Client must be in the search page of the PowerEnjoy app
- Client must click the manage profile button

Flow of events:

- If the client press edit password:
  - PowerEnjoy redirects him to the change password page
  - The client enters the old and the new password in the fields
  - PowerEnjoy checks the old password
  - PowerEnjoy saves the new password
- If the client press edit email:
  - PowerEnjoy redirects him to the change email page
  - The client enters the old and new email in the fields
  - PowerEnjoy check if the old email is right
  - PowerEnjoy checks if the new email doesn't belong to other client
  - PowerEnjoy saves the new email
- If the client press edit payment information:
  - PowerEnjoy redirects him to the change payment information page
  - The client enters the new payment information in the field
  - PowerEnjoy checks if the new payment information is valid
  - Power Enjoy saves the new payment information
  - If the client account was suspended, PowerEnjoy actives it again

Exit condition: no exit condition

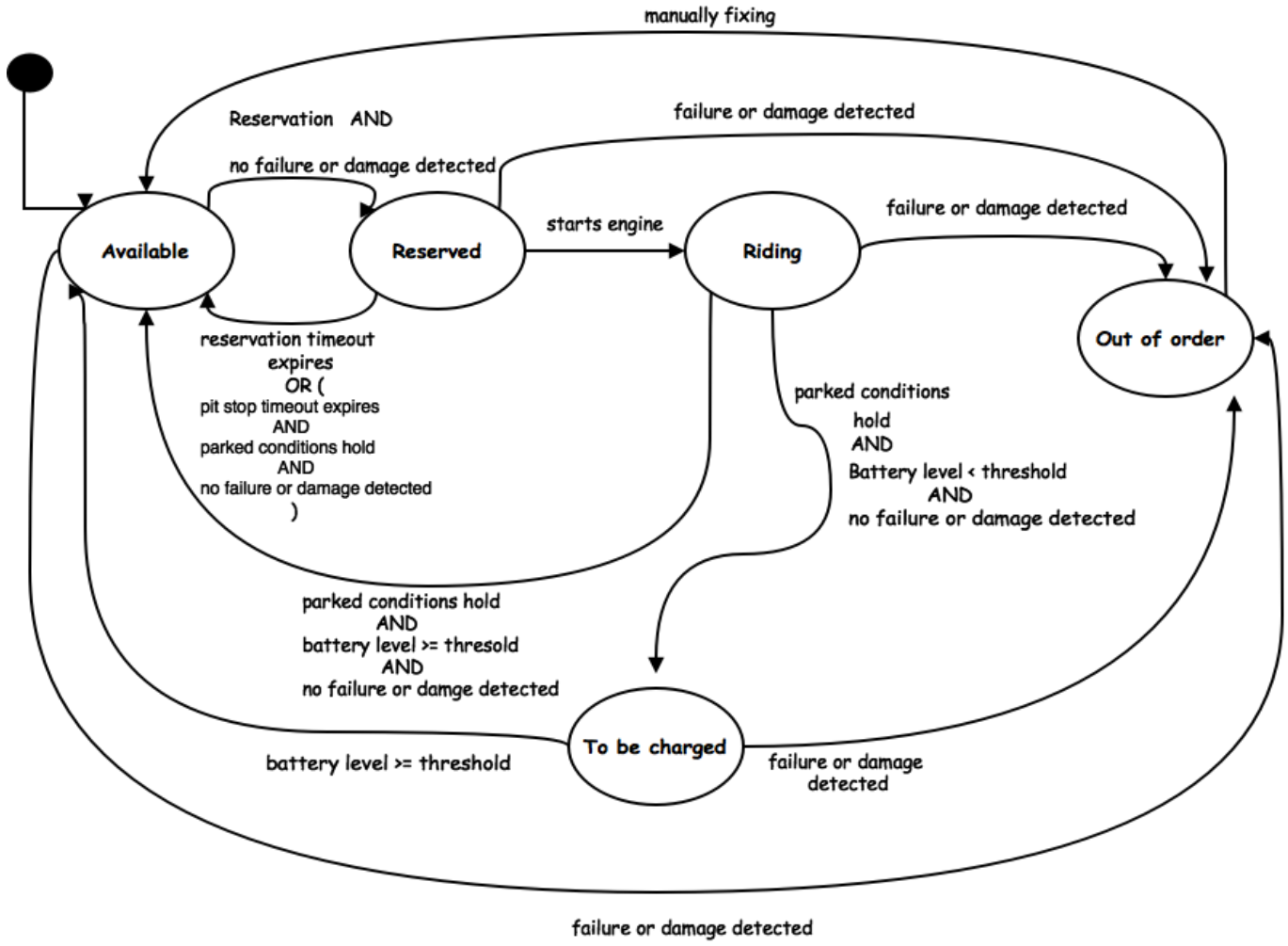
Sequence diagram:



5.3 State machine diagram

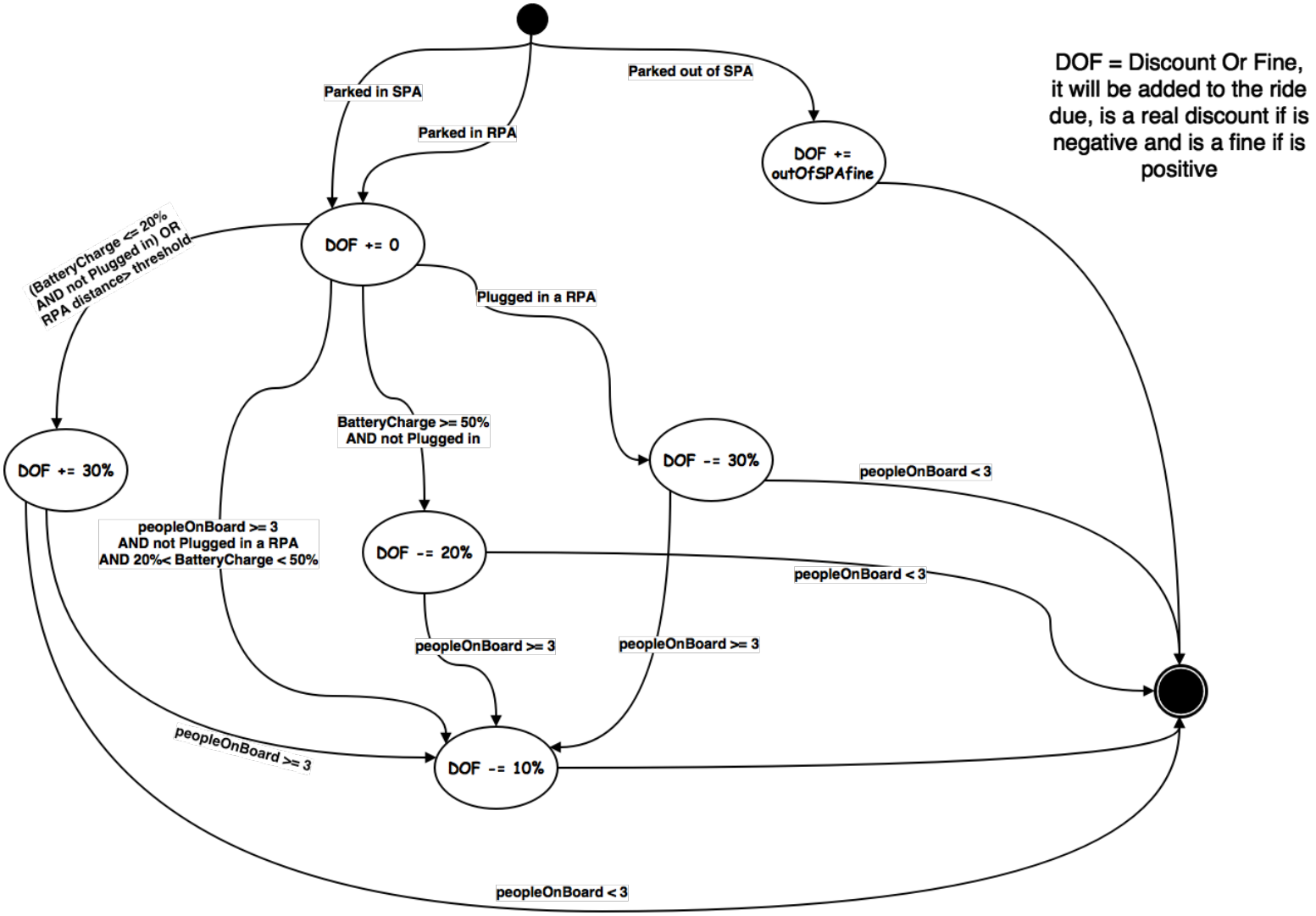
5.3.1 State diagram 1

This diagram shows the possible statuses of a car in PowerEnjoy.

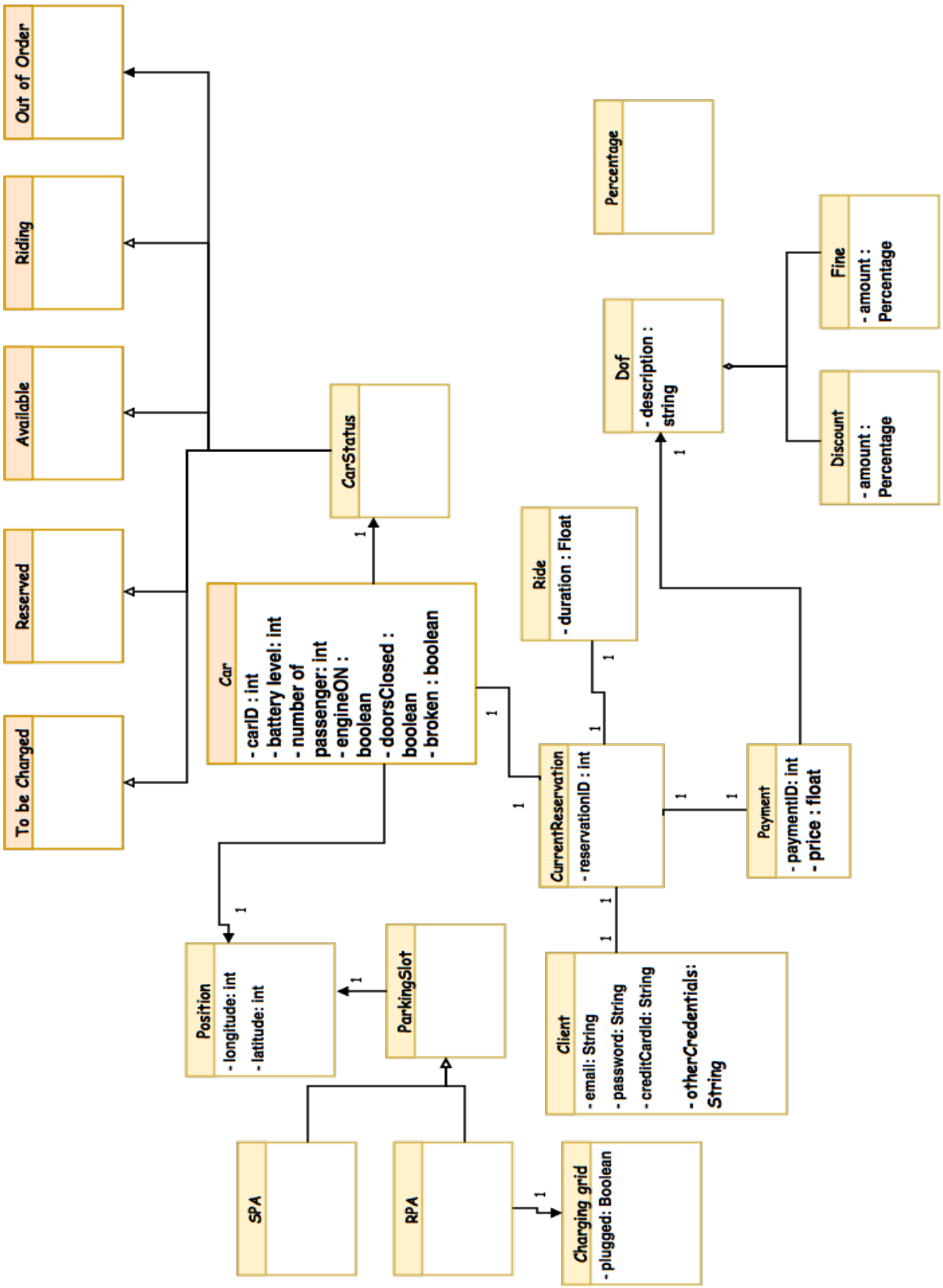


5.3.2 State diagram 2

This diagram shows the possible fine and discount that PowerEnjoy record. Moreover, it shows the way in which they are summed.
The DOF is the sum of the percentage and it is applied to the total ride.



5.4 Class Diagram



6 Appendix

6.1 Alloy Modelling

6.1.1 Signatures

```
open util/boolean

sig String1 {}

sig Car{
	carId: one Int,
	battery_level: one Int,
	numb_of_passengers: one Int,
	engine_ON: one Bool,
	doorsClosed: one Bool,
	broken: one Bool,
	status: one CarStatus,
	position: one Position
}
{
numb_of_passengers>=0
battery_level>=0
}

abstract sig CarStatus{}
one sig ToBeCharged extends CarStatus{}
one sig Reserved extends CarStatus{}
one sig Available extends CarStatus{}
one sig OutOfOrder extends CarStatus{}
one sig Riding extends CarStatus{}


sig Client{
	email: one String1,
	password: one String1,
	creditCardId : one String1
}
```

```
sig CurrentReservation{
	reservationId : one Int,
	car: one Car,
	ride: lone Ride,
	client: one Client,
	payment: one Payment

}

sig Ride{
	duration: one Int
}{
duration >= 0
}

sig Payment {
	paymentId : one Int,
	price : Int,
	dof: lone Dof
}
{
price >0
}


sig Dof {
	description: one String1,
	discount: set Discount,
	fine: set Fine
}{
	#discount >= 0
	#discount < 4
	#fine >= 0
	#fine < 3
}

sig Discount{
	amount:one Int
}
```

```
sig Fine{
	amount: one Int
}

sig Position{
	longitude: one Int,
	latitude: one Int
}

sig ParkingSlot{
	position: one Position
}

sig Spa extends ParkingSlot{}

sig Rpa extends ParkingSlot{
	chargingGrid: one ChargingGrid
}

sig ChargingGrid{
	plugged: one Bool
}
```

6.1.2 Facts

```
fact differentCarsCantHaveSameIDs{
        all c1,c2: Car | c1!=c2 <=> c1.carId != c2.carId
}

fact differentCurrenReservationsCantHaveSameIDs{
        all r1,r2: CurrentReservation | r1!=r2 =>
        r1.reservationId !=    r2.reservationId
}

fact differentCurrentReservationsMeansDIfferentRides{
        all c1,c2:CurrentReservation |   c1!= c2 <=> c1.ride    != c2.ride
}

fact rideExistsOnlyConnectedWithAReservation{
        all r:Ride |  some c: CurrentReservation | c.ride = r
}

fact differentCurrentReservationsMeansDIfferentCars{
        all c1,c2:CurrentReservation |   (c1!= c2 <=> c1.car != c2.car)
}

fact differentReservationsMustHaveDifferentClients{
        all c1,c2:CurrentReservation |   (c1!= c2 <=> c1.client != c2.client)
}
```

```
fact differentClientsMustHaveDifferentEmails{
	all c1,c2 : Client | (c1!= c2 <=> c1.email != c2.email)
}

fact differentReservationsMustHaveDifferentPayment{
all c1,c2:CurrentReservation |   (c1!= c2 <=> c1.payment != c2.payment)
}

fact differentPaymentsMustHaveDifferentIds{
	all p1,p2 : Payment | ( p1!= p2 => p1.paymentId != p2.paymentId )
}

fact paymentsExistOnlyWithAReservation{
	all p : Payment | one c : CurrentReservation |  c.payment = p
}

fact dofExistOnlyAttachedToAPayment{
	all d: Dof | one p : Payment | p.dof = d
}

fact fineExistOnlyWithADof {
	all f: Fine | one d:Dof | d.fine = f
}

fact disocuntExistOnlyWithADof {
	all d: Discount | one df:Dof | df.discount = d
}
```

```
fact differentCarsHaveDifferentPositions{
	all c1,c2 : Car | c1!=c2 => c1.position != c2.position
}

fact differentPostionHaveDifferentLatitudeorLongitude{
	all p1,p2 : Position | p1 != p2 => ((p1.latitude != p2.longitude ) ||  (p1.longitude != p2.longitude))
}

fact positionExistOnlyWithACarOrParkingSlot{
	all p:Position | one c:Car |  c.position = p
}

fact differentParkingSlotHaveDifferentPosition{
	all ps1,ps2: ParkingSlot | ps1!= ps2 => ps1.position != ps2.position
}

fact differentRpaHaveDifferentChargingGrid{
	all r1,r2 : Rpa | r1!= r2 => r1.chargingGrid != r2.chargingGrid
}

fact chargingGridsExistOnlyWithRpa{
	all c: ChargingGrid | one r: Rpa | r.chargingGrid = c
}
```

```
fact parkedConditions{
	all c: Car  | (c.status = Reserved) => ( c.engine_ON = False && c.broken = False && c.battery_level >= 2 && c.numb_of_passengers = 0 && c.doorsClosed = True )
}

fact reservedCarsAreReserved {
	one r:CurrentReservation | all c: Car | (c.status = Reserved) => (r.car = c  )
}

fact carsWithLowBatteryAreToBeCharged {
	all c: Car | (c.status = ToBeCharged => (c.battery_level < 2  && c.broken = False && c.engine_ON = False && c.numb_of_passengers = 0 && c.doorsClosed = True  ) )
}

fact toBeChargedCarsDontHaveACurrentReservation{
	all r:CurrentReservation | no c: Car  | (c.status = ToBeCharged ) &&  r.car = c
}

fact brokenCarsAreOutOfOrder{
	all c:Car | (c.broken = True) <=> (c.status = OutOfOrder)
}

fact OutOfOrderCarsDontHaveACurrentReservation{
all r:CurrentReservation | no c: Car  | (c.status = OutOfOrder ) &&  r.car = c
}
```

```
fact ridingCarsAreRiding {
        all c:Car | ( (c.status = Riding ) => (c.engine_ON =
True  &&  c.broken = False && c.battery_level >0) )
}

fact ridingCarsHaveACurrentReservation {
one r:CurrentReservation | all c: Car | (c.status = Riding) => (r.car = c)
}

fact AvailableCarsDontHaveACurrentReservation{
all r:CurrentReservation | no c: Car  | (c.status = Available ) &&  r.car = c
}

fact AvailbleCars{
            all c:Car | c.status = Available => ( c.engine_ON
= False && c.broken = False && c.battery_level >= 2 &&
c.numb_of_passengers = 0 && c.doorsClosed = True )  }
```

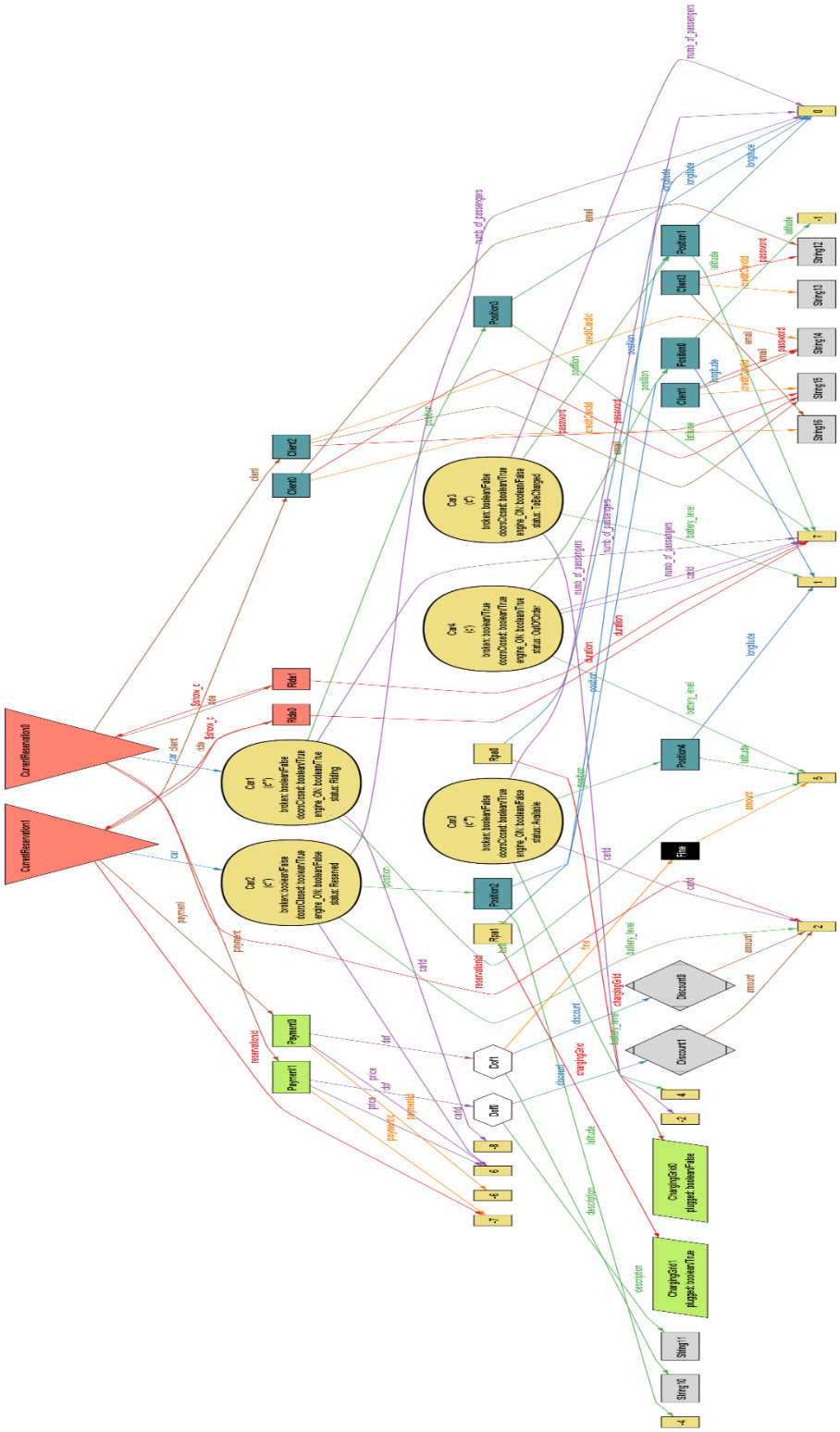
6.1.4 Assertions

```
assert AllReservationsHaveAReservedOrDrivingCar{
        all r : CurrentReservation | no c:Car | (c.status != 
Reserved &&  c.status != Riding ) && r.car =c
}
check  AllReservationsHaveAReservedOrDrivingCar for 5



assert ClientCantReserve2CarsAtTheSameTime{
        all r1,r2 : CurrentReservation | (r1.client != r2.client 
<=> r1.car !=r2.car)
}
check ClientCantReserve2CarsAtTheSameTime for 5



assert DifferentClientsMustHaveDifferentPayments{
                all r1,r2 : CurrentReservation | r1.client != 
r2.client <=> r1.payment != r2.payment
}
check DifferentClientsMustHaveDifferentPayments for 5
```

6.1.5 Predicates

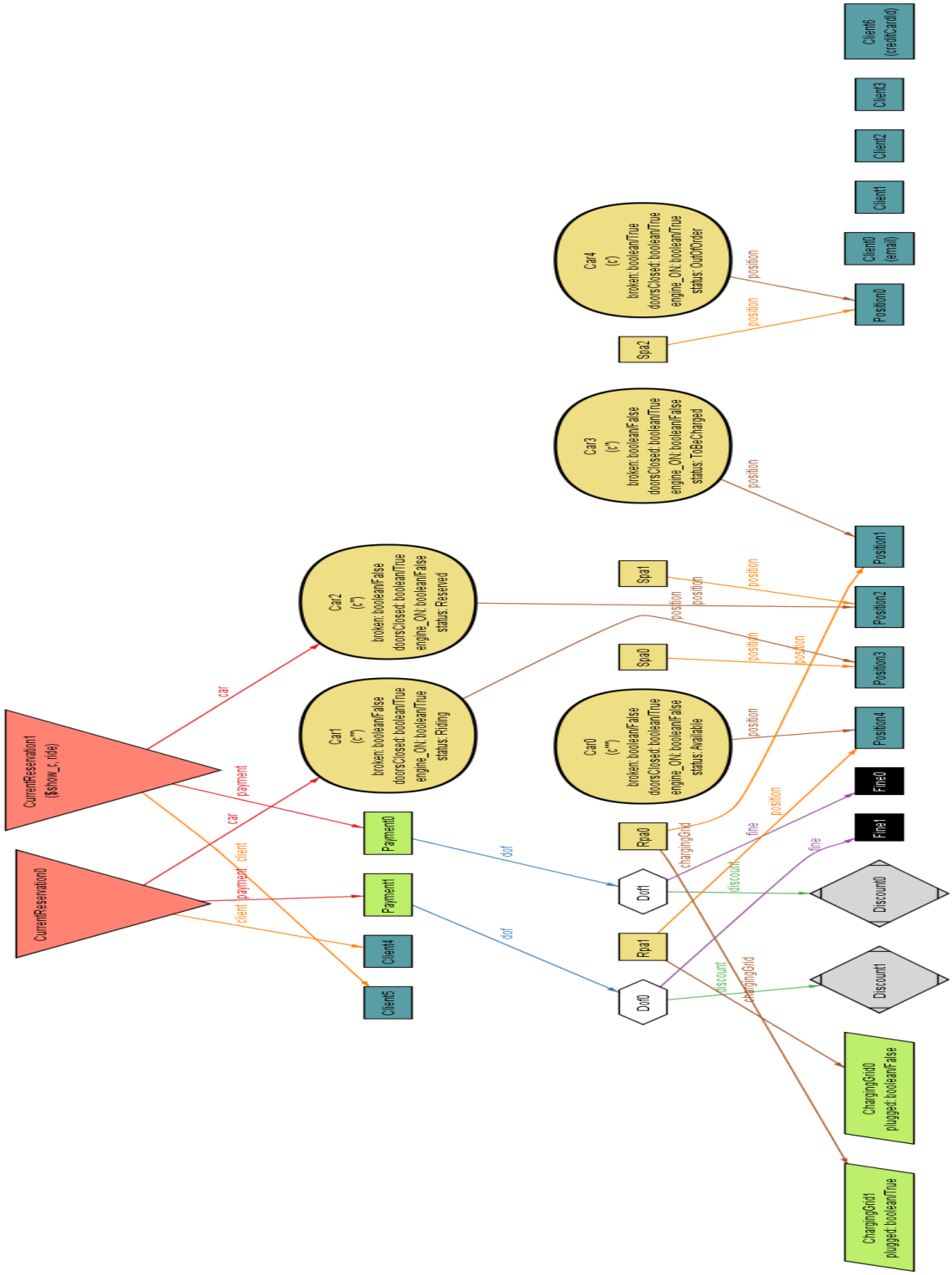
```
pred show{
        #Position = 5
        #Car >2
        #Rpa = 2
        #ChargingGrid = 2
}
run show for 7
```

6.1.6 Generated Word



A clearer image of this world is attached to this document

Generated world with some projections to make it simpler



6.2 Used Tools

- **Dropbox**
- **Microsoft Word**
- **www.draw.io** for class diagram and state diagram editing
- **www.websequencediagrams.com** for sequence diagram editing
- **Alloy Analyzer** to prove the consistency of the model

6.3 Hours of Work

- Alessandro Terragni 53h
- Alessandro Polenghi 49 h

6.4 Change Log

- v1.1
  - removed map management and driving license management from CMS
  - modified The World and The Machine Diagram: System Logic and CMs added, RPA and SPA moved to shared phenomena
  - add a clearer image of alloy world
  - add lanes to sequence diagrams
  - fix in 'Park the car' sequence diagram
  - fix in 'Reservation' sequence diagram