POLITECNICO
MILANO 1863

PROGECT for
SOFTWARE ENGINEERING 2

Requirements Analysis and Specifications Document

Politecnico di Milano
A.A. 2016-2017
Prof.ssa Elisabetta Di Nitto

Students:
Diego Gaboardi
Giorgio Giardini
Riccardo Giol

INDEX

1 INTRODUCTION

1.1 DESCRIPTION OF THE GIVEN PROBLEM

Our project is about PowerEnJoy, a system of car-sharing that employs only electric cars, operating on the territory of Milan. It provides users to reserve and use shared cars of the system in a specific area, paying in function of the time of driving. The interface to communicate to the system is a mobile app that make the renting easy and clear. After reserving a car and getting on it with a few taps on the smartphone app, a monitor in the car helps the driver with some available commands and suggestions, and shows the amount of money that has to be paid in real time. All the interactions between the user and the system are managed by the mobile app and the monitor on the car. They guide the user to the correct use of the system.

To be able to use the service the user has to associate his driver’s licence and a valid credit card to its profile, with which he will pay the travel. Every time the user wants to use the service he has to insert his credentials. Through the app he is able to get in the car. There are 5 minutes for the driver to get himself comfortable in the car after which the payment will start. At the end of the travel the driver has to leave the car in a safe area, and the payment will be automatically charged to the user’s credit card.

Furthermore, he can respect some constraints to get some bonuses on the rate. In fact, the system wants to reward virtuous users with some discounts: for example, if they transport other people or leave the car in a recharging area, they manifest a particular attention at the environment and at the system.

Repositioning of the cars, their recharging and their reparation are managed by a group of assistants that can know and modify, through their special account, the position of each vehicle and the entity of its problem. With a special *passe-partout* they can use cars freely in order to offer a good and equally distributed service.

1.2 GOALS

- [G1] Clients are allowed to register to the system giving their credentials and payment information;
- [G2] Registered clients are able to see through an interactive map the positions of the available cars near a specific address (current position or inserted);
- [G3] Registered clients can reserve a single electric car for at most an hour before picking it up;
- [G4] Clients that get the reservation countdown expired are punished with a fee of one euro;
- [G5] Clients can open the reserved car scanning the QR code;
- [G6] Clients can monitor the amount of money to be payed, updated in real time on car display;
- [G7] Client can enable the saving option in order to ensure a uniform distribution of car in the city;
- [G8] Clients can leave the car locked in break continuing to pay, keeping it reserved;
- [G9] Cars are blocked automatically when client ends his travel;
- [G10] Client are charged proportionally to the driving time with some penalties or discounts;
- [G11] Specific employee can register at the system with a special account, becoming Assistant;
- [G12] Assistants can know the position and the state of all electric cars;
- [G13] Assistant can change the state of a car;
- [G14] Registered clients can modify their account, such as their credit card.

1.3 DOMAIN PROPERTIES

These are the main properties of our world that need to be always true:

- GPS always give the right position;
- The service is always available;
- User cannot switch off the car information system;
- The connection of car information system never fail;
- Course time is always positive;
- The sensors system in each car always detects a positive number (or zero) of clients;
- Countdowns cannot be stopped;
- Car’s QR codes are unique;
- A car can only be in one place at a time;
- Broken cars are always not available;
- Assistants makes cars available only when they are effectively in good conditions;
- In case of accident, our insurance is able to operate directly with the client, making use of data provided by us;
- Drivers never stop in the road with 0% of battery;
- Drivers do not come with the car out from the available areas;
- All cars attached to a power plug are in charging state;

1.4 GLOSSARY

Registered client: he is a client who have already done the procedure for signing in. He provided to the system all the necessaries data: his credentials and payment information. He has to own a valid driving licence.

Reserving client: he is a "registered client" who reserved a single electric car through PowerEnJoy. He has at most one hour for picking it up.

Driving client: he is a "registered client" who is actually in the car.

Dismounted client: he is a "registered client" who doesn't have any reservations or active course on PowerEnJoy. Then he is allowed to reserve available electric cars.

Client in break: he is a "registered client" who exits the car, but decides to keep it reserved. For this reason, he continues to pay for the car.

Electric car: it is an automobile owned by PowerEnJoy that is propelled by one or more electric motors, using electrical energy stored in rechargeable batteries.

Available car: it is an "electric car" which currently doesn't have any reservations. It is in a safe area with more than 20% of battery and in good conditions. Then registered clients can reserve them.

Reserved car: it is an "electric car" which is currently reserved by a "reserving client". Therefore, in this state it cannot be reserved by other clients.

Car in break: it is an "electric car" which is locked and turned off. It continues to be reserved and to charge the client.

Reservation countdown: it is a one-hour countdown that starts when the client reserves an available car and ends when he picks it up. During this time the car continues to be considered a "reserved car".

Courtesy countdown: it is a five minutes countdown that starts when the client opens the car and ends with the engine ignition. In this period the client, before starting to drive, can get himself comfortable, put in a proper positions his bag, adjust car mirrors and fast security belt.

Current position: it is the position detected by the GPS of the client device.

Virtuous behaviour: a way in which a person behaves respecting the environment. Examples of virtuous behaviours are sharing the car with other passengers or leaving the car in the special parking areas where they can be charged.

Zone: it is a zone of approximately 1km^2. The town is divided in several zones.

Available constraints: the car must have more than 20% of battery, be in good conditions (clean and without damages) and locate in a "safe area".

Safe area: it is any parking accessible through an electric car located in the territory of Milan and stored in the system database.

Special parking area: it is a “safe area” in which there are plugs and so electric cars can be recharged.

Available area: it is an area of the Milan territory in which “electric car” can go.

Power plugs: plugs adequate to recharge the electric car. They have a sensor that detect if it is connect to a car or not.

Assistant: he is an employee with a special account that enable him to use freely the cars for doing their job

Green number: it is the number that the client calls in order to communicate a fault. It is written clearly visible on each car

1.5 TEXT ASSUMPTIONS

There are few points that are not very clear in the specification document, or not completely specified. Therefore, we will have to make some assumption and choices:

- The system is able, in the moment of the registration, to assure that the provided credentials are consistent and the client has a regular and active driving licence. This is true also for payment information: only credit (not debit) cards are accepted and the system is able to verify with principal bank circuit (ex. Visa or Mastercard) the actual validity of the card itself.

- The method for opening cars is to scan a QR code that each car has on its left door.

- In order to avoid people entering the car and stationing inside without paying, we put in the system a 5 minutes countdown, starting from the moment in which client enters the car. After these 5 minutes, user starts to be charged, even if the engine is switched-off.

- All the cars used by our clients are provided with weight sensors, one for each possible passenger. It is the way in which our system is able to count passengers and verify if someone is still in the car.

- The discounts are combinable. If a client reaches more than one objective, he will be discounted subsequently from the highest to the lower discount.

- In order to assure an always active service and an equal distribution of the car in the city the society must assume some assistants. They can access to a special area of the application, seeing where all cars are placed and their current state. This allows them to turn constantly around the city, recharging cars with their special equipment and redistributing vehicle in the city. Assistant can't open cars by application, they are equipped with a *passepartout.* After recharging or repairing a fault they are able to change car state, making it again available.

- Client that detects a fault in the car calls PowerEnjoy green number. An assistant, with the special temporary role of switchboard operator, thanks to client statements, change car state and insert the entity of the fault.

- Safe areas stored in the system are all the regular 'white line' parking in the city of Milan. The system doesn't enable clients to leave the car parked and finish the course in a not-safe area. The system is able to verify the matching between a GPS position and a position registered in PowerEnjoy database, however it can’t assure that the client has parked the car correctly into white box of the parking. Eventual problems can be notified at the system by its green-number.

1.6 COSTRAINTS

- REGULATORY POLICIE
  Sensitive data such as credentials, information about payment and movements of the client must be used respecting the privacy law.

  Furthermore, the system, in order to work correctly, must require the client permission to get his position.

  Finally, notifications (not SPAM) can be sent to the client for exchanging necessary information between the application and the system.

- HARDWARE LIMITATIONS
  - Mobile application:
    - 3G connection
    - GPS
    - Space for app package
    - Photo camera for scanning QR code
  - Car application:
    - 3G connection
    - GPS

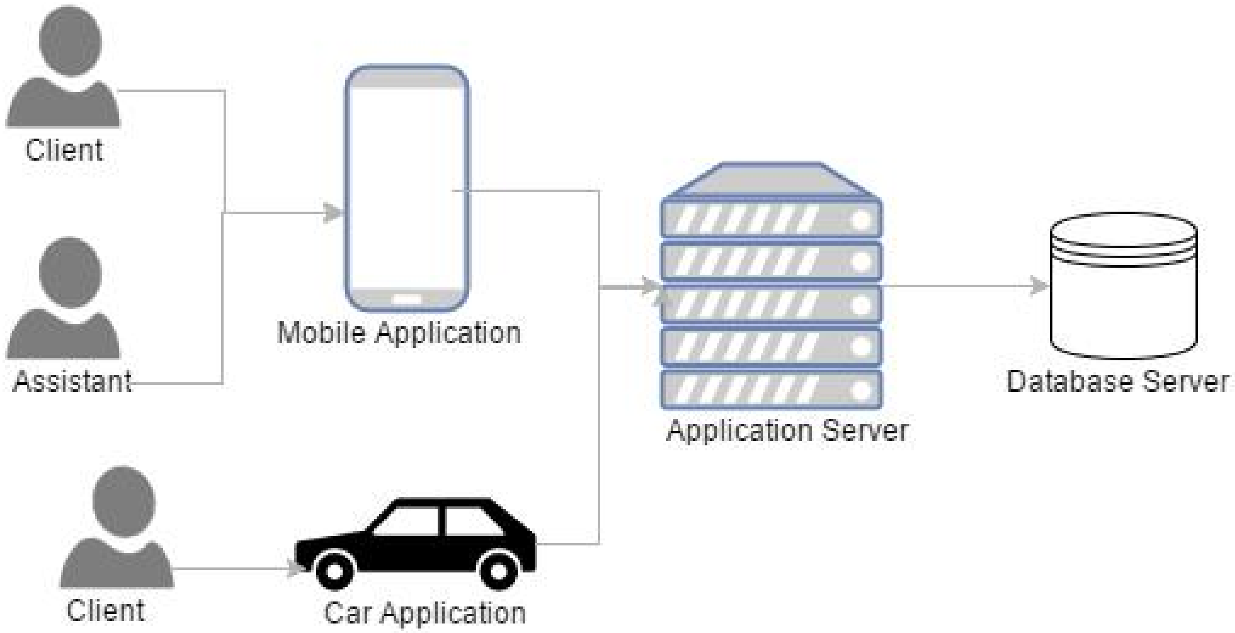
- INTERFACES TO OTHER APPLICATIONS
  The system will interface (through appropriate APIs) with the phone camera, in order to scan QR codes necessary for opening the car.

  The system will also interface with an entity of online payment (such as PayPal) in order to execute transaction between clients and PowerEnjoy society.

  Finally, is essential, both for client and assistant, to be able to visualize cars on a city map. For this reason, the system must interface with a map service (such as Google Maps).

- PARALLEL OPERATIONS
  The system must manage the interactions with several different client applications at the same time. For this reason, it must handle concurrent operations.

- PROPOSED SYSTEM
  We decided to implement a client-server architecture in which there is a mobile application that interacts with an application server. This server receives essential information also from a different application situated in the car. At the same time, it stores and uses information in a database server. We will speak about architecture more precisely in the following documents.


Client
Assistant
Client
Mobile Application
Car Application
Application Server
Database Server

1.7 IDENTIFYING STAKEHOLDERS

Our project is commissioned by the minister of the transport of Milan in agreement with the Minister of the Environment. They request the project documentation until the end of February. The documentation consists in the Requirements Analysis and Specification Document for have a general and complete idea of the system in exam, the Design document for have a functional description of the system, the Integration Test Document that describes how to accomplish the integration tests and the Project Plan.

With this system, the minister of the environment wants to sensitize the inhabitants at a sustainable system for how concerns the transport. In the system, moreover, are introduced some bonuses for the people who respect some constraints, for reward their virtuous behaviour.

1.8 ACTORS IDENTIFYING

The actors involved in the system, are:

- Client: he is registered in the system and logged when we mention him in this document. He uses the system for personal purpose. He uses the app in his smartphone, and can offer the travel to other people at the same time, according to the capacity of each car.
- Assistant: he is an employee that have a special account that enable him to move the car without paying. He is delegate to redistribute cars in the available areas and recharge cars with low battery.

1.9 REQUIREMENTS

1. Clients are allowed to register to the system giving their credentials and payment information.
    a. The system is able to control if credentials are correct;
    b. The system is able to control if payment information are correct;
    c. The system is able to confirm registration and send a password to the new client.

2. Registered clients are able to see through an interactive map the positions of the available cars near a specific address (current position or inserted).
    a. The system is able to get the position from the GPS or from user input;
    b. The system knows the actual position of all electric cars;
    c. The system can generate a map with a marker for each available car in the selected zone.

3. Registered clients can reserve a single electric car for at most an hour before picking it up.
    a. The system is able to modify the state of the electric car (in this case from "available" to "reserved"). This is necessary also for content goals 4-5-8-9;
    b. The system is able to modify the client state (in this case from "dismounted" to "reserving"). This is necessary also for content goals 4-5-8-9;
    c. The system is able to remember which client reserves which car by storing a relationship between the car and the client;
    d. The system is able to keep in mind the time remaining for the client to take the car by setting a 1-hour reservation countdown.

4. Clients that get the reservation countdown expired are punished with a fee of one euro.
    a. The system is able to verify when countdown expires;
    b. The system is able to charge the client with a fee of one euro;
    c. The system closes the reservation when countdown expires.

5. Clients can open the reserved car scanning the QR code.
    a. The system is able to control the correspondence between the scanned QR code and reserved car;
    b. The system is able to unlock the reserved car;
    c. The system is able to start a 5 minutes courtesy countdown in the moment in which the client unlocks the cars (when the countdown expires the system starts to charge the user even if engine is still switched off).

6. Clients can monitor the amount of money to be payed, updated in real time on car display.
    a. The system calculates the amount of money that has to be paid by the client starting from engine ignition (or courtesy countdown end);
    b. The amount of money is displayed on car monitor and is continuously updated;
    c. The system stops charging the client at travel conclusion.

7. Client can enable the saving option in order to ensure a uniform distribution of car in the city.
    a. The system is able to find the geographical position of the final destination inserted by the client;
    b. The system is able to calculate distances between the destination and the nearest special parking areas;
    c. The system always knows how much power plugs are available in the parking areas that have been found;
    d. The system is able to find the best special parking area for the client basing on information acquired in point 7b and 7c.

8. Client can leave the car locked in break continuing to pay, keeping it reserved.
    a. The system is able to detect the absence of passengers on board. This is necessary also for content goal 9;
    b. The system is able to ask the client if he wants to end his travel or to leave the car in break, sending him a notification through the app. This happen when sensors detect that all passengers are out of the car, engine is turned off, doors are closed and car position match with a safe area.
    c. The system is able to lock the car keeping it reserved by the client who is on break;
    d. The system is able to unlock the car when the client rescans the QR code through the application after the break.

9. Cars are locked automatically when client ends his travel
    a. The system is able to ask the client if he wants to end his travel or to leave the car in break, sending him a notification through the app. This happen in the same way of point 8.b;
    b. The system is able to lock the car;
    c. The system is able to verify if the vehicle respects the available constraints, consequently modifying its state.

10. Client are charged proportionally to the driving time with some penalties or discounts.
    a. The system is able to verify the distance between a parked car and the nearest special parking area and the remained level of battery, applying a 30% penalty in case the distance is more than 3 km or the level is less than 20%;
    b. The system is able to verify how much passengers have been part of the ride, applying a 10% discount in case they are more than 3;
    c. The system applies a 20% discount on last ride in case that battery level is more than 50%;
    d. The system is able to detect if the car has been plugged by the client into a power grid, applying in this case a special discount of 30%;
    e. The system, that knows client payment information, is able to execute the transaction.

11. Specific employee can register to the system with a special account, becoming assistant.
    a. The system is able to check if credentials inserted by the user are relative to an employee regularly hired by society;
    b. The system is able to confirm registration of the special account and send a password to the new assistant.

12. Assistants can know the position and the state of all electric cars.
    a. The system knows the actual position and the state of all electric cars registered;
    b. The system is able to generate a map including each car state and position, showing it to the assistant with an interactive map.

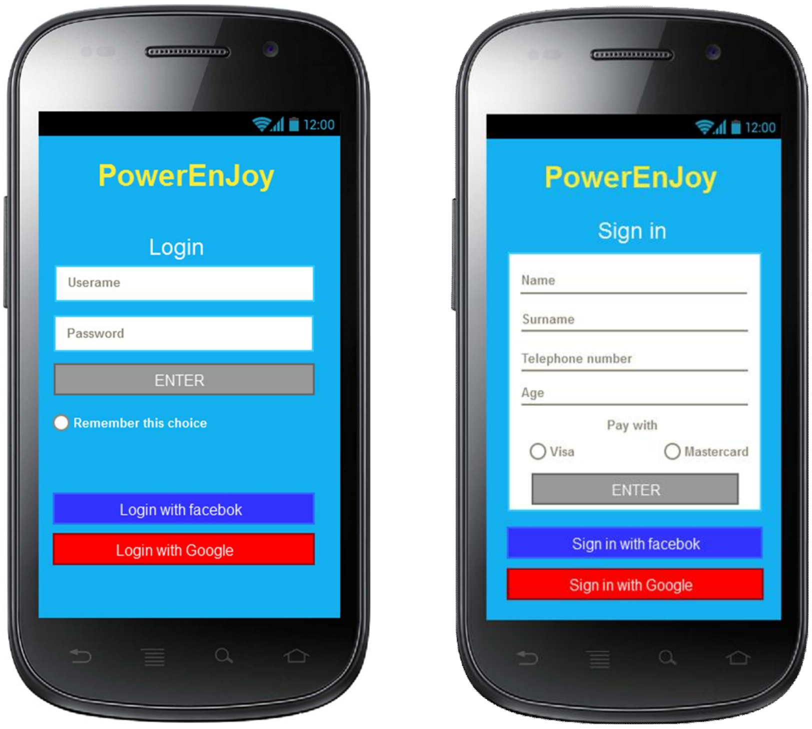
13. Assistant can change the state of a car.
    a. The system allows the assistant to select a car in the interactive map;
    b. The system is able to change the state of the car selected respecting assistant decision.

14. Registered clients can modify their account information, such as their credit card
    a. The system can modify the interested client data in the database;
    b. The system can modify the credit card associate to an account, verifying its validity.

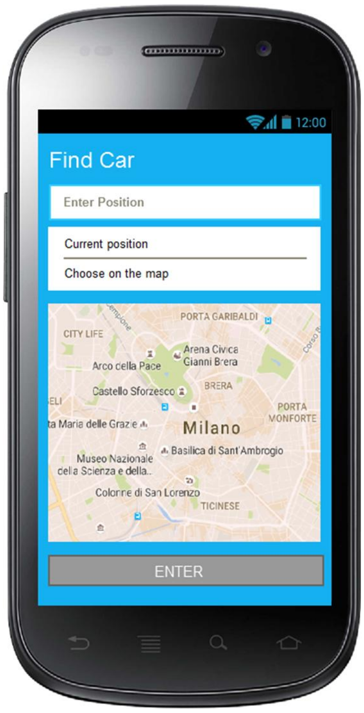
1.10 NON FUNCTIONAL REQUIREMENTS

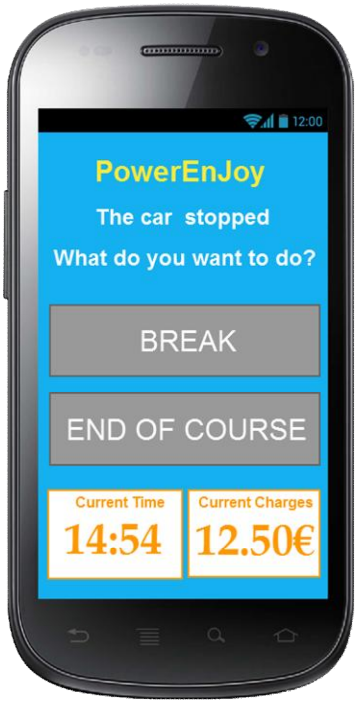
- Service always available, 24 hours a day and 7 days a week.
- Client and car state updated in the order of 1 second.
- Client information and payment data are reserved.
- System must be able to support multiple connections at once.
- Client interface is intuitive and easy to use.
- All transactions, such as reservation and payment, must be atomic.

Below we report some examples of how we would like our interface to be



These two pictures show the login and sign in page





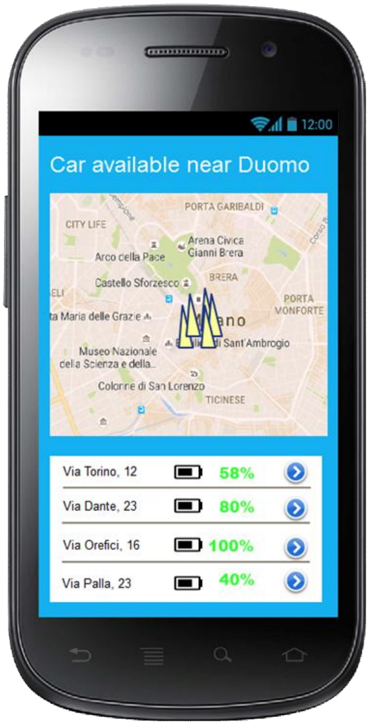
The car stopped

What do you want to do?

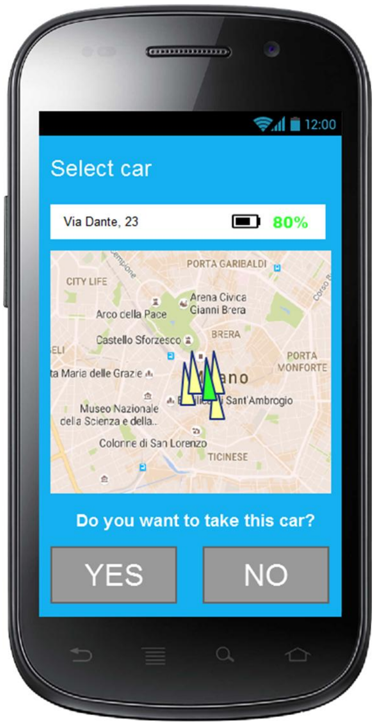
Current Time	Current Charges
14:54	12.50€

The first picture shows the page in which a client can insert the position where he wants to take the car.

In the second one there is the notifications in which clients can decide if end their travel or leave the car in break.



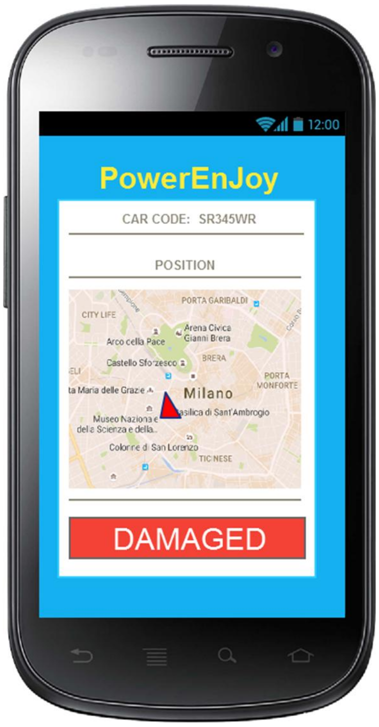
Via Torino, 12	58%
Via Dante, 23	80%
Via Orefici, 16	100%
Via Palla, 23	40%



Via Dante, 23 80%

Do you want to take this car?

The first picture shows the cars available near Duomo. For each car, there is the battery percentage and a button for selecting the car and going to the section represented in the right figure in which client are asked to confirm their choice.

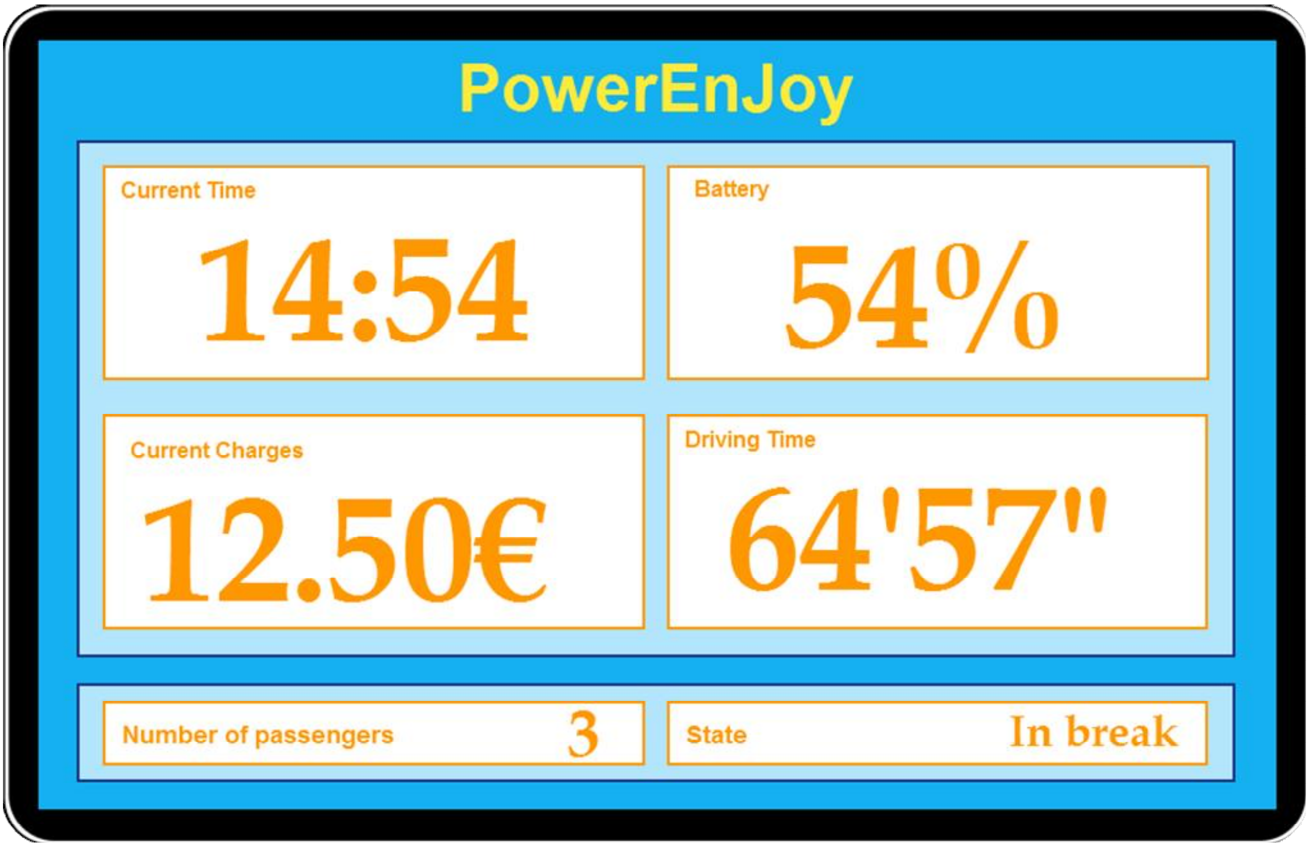


PowerEnJoy

CAR CODE: SR345WR

POSITION

This picture shows the special area of the application relative to the assistant in which he can control car states.



PowerEnJoy

Current Time 14:54

Battery 54%

Current Charges 12.50€

Driving Time 64'57"

Number of passengers 3

State In break

This pictures shows the car application in which clients can see the battery, the driving time, the current charges and time continuously updated.

1.11 SCENARIO IDENTIFYING

SCENARIO 1

Jack and his girlfriend Mary are two students that decided to get a trip. They bought the cheapest fly at 5, very early in the morning. To reach the airport they have to take the train from central station. At this hour in the morning there are not public transport, so Jack decides to try to use PowerEnJoy, that assures the service 24 to 24, 7 to 7. After having booked a car full recharged in the special parking area nearest to his home, Jack goes with his baggage to the vehicle. There he opens the car with his smartphone, scanning QR code through the application, gets in, sets the seat in the right position and straights mirrors. So, he goes to Mary house. Once arrived, Jack parks the car in a safe area in break state, selecting the correspondent option on his smartphone, and goes up to Mary`s apartment in order to help her with her baggage. After having again scanned QR code and unlocked the car, they all get in and drive to the central station where another special parking area is placed. Once arrived to the station and took off their baggage, Jack close the doors and confirms on his smartphone the end of the race. Only in this moment he stops paying and the system executes the transaction from his credit card.

SCENARIO 2

Tonight Bob wants to go to a disco with 2 friends, who live in the same city but in different districts. This evening is cold and it's raining so they don't want to take public transport. As they want to save some money, Bob, who has heard of this service from other friends purposes them to rent a PowerEnJoy car, instead of taking a taxi. In fact Bob knows that, sharing the ride, they will get a special discount. In the afternoon Bob registers to the application and then, at 21.00 he logins and reserves the nearest car to his house. At 21.30 he goes out and takes his car before that the 1 hour countdown expires. After taking his friends, when they are almost near to the disco, Bob is very happy to find through the application a special parking area just next to the discotheque. Indeed, he know that by leaving the car in recharging state they will be able to increase their special discount. Therefore, they reach the area and, after having put the car in charge, they select the park option from the application. The system applies the discount and executes the transaction. Now Bob and his friends can enjoy their night.

SCENARIO 3

Mohammed, after being looking for a job for many years, has finally found an employment as assistant in PowerEnJoy enterprise. He is a fundamental part of the system. In fact, today, an unvirtuous client has left the car even 6 km far from the nearest special parking area and with only 5% of power left. No one will be able to use a car in this condition. For this reason, the car is set in a no available state and the client has been charged with a penalty of 30% on the total amount of his last ride. So Mohammed logins to the system with his special employee account and here he can see both the state and the position of all not available cars. Once found the mentioned vehicle, he goes immediately on the site, recharges the car on site with his special equipment, opens the car with his *passepartout* and moves the car to the nearest recharging area. The system identifies the recharging car and when the vehicle reaches at least 50% of battery is able to change again its state and set it available. Mohammed can now open his map again in order to find another car to move.

SCENARIO 4

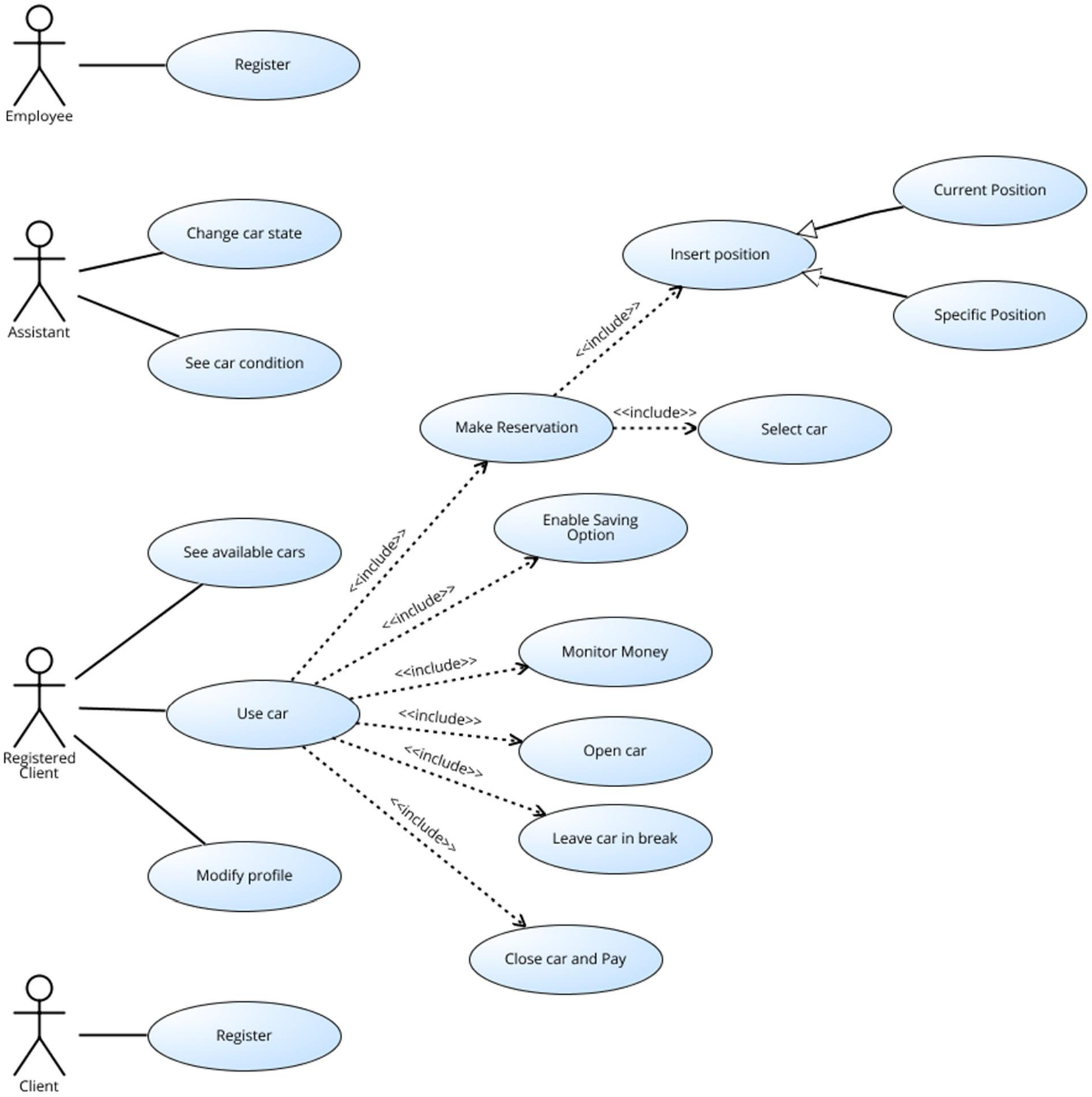
Steve is a 18 years old boy, who has obtained his driving license just last week but doesn't have an own car yet. Today Steve wants to reach his girlfriend Anna on the opposite side of the city and in order to preserve the environment he decides to use PowerEnjoy electric car. Once inserted his personal credential and credit card data he is finally registered to the system and he can reserve his car. By reading the rules of the service he comes to know of the money saving option and decides to try this advanced function. So he gets in the car and before starting his travel he inserts in the system Anna's address. After few seconds the app shows him that there is a recommended parking area at about 1km far from his girlfriend's house with lots of power grid free. In order to save some money Steve decides to follow the advice and gets to the place in about 15 minutes. Once arrived and plugged the car into the grid, the system applies him a special discount of 30% on the price of the ride. Steve, glad for this discount, promises himself to use again this convenient functionality offered by PowerEnjoy application.

SCENARIO 5

For Christmas Rosy has cook some biscuits for her friends. She wants to make a surprise, so decides to pass at midnight to deliver the presents. To pass to all her friends, inhabiting in different quarter of the city, she decides to use an electric PowerEnJoy car, because it's too cold for reaching them by bicycle and to expansive using a taxi. Through the service Rosy can successfully stops and leave the car in break at each friend's houses, always respecting the safe area constraints, spending a lot less then she would have spent using a taxi.

2 UML MODELS

2.1 USE CASE DIAGRAM



2.2 USE CASE DESCRIPTION

In the situation described in the previous section we found the following use case:

- Register a client;
- Use a car;
- Make a reservation;
- Insert position;
- Insert current position;
- Insert specific position;
- Select a car;
- Enable saving option;
- Monitor money;
- Open the car;
- Leave the car in break;
- Close the car and pay;
- See available cars;
- Modify client profile;
- Register an employee;
- Change car state;
- See car state.

Now we are going to analyse most significant use cases more precisely. At this point we refer to “input”, “pages” or “button” only as hypothesis to explain all the situations in a clearer way. However, the real structure of our system will be defined in the Design Document.

NAME	REGISTER A CLIENT
Actors	Client
Entry conditions	The client has downloaded and installed PowerEnjoy application. The client has all data requested for the inscription.
Flow of events	The client asks for registration. The client inserts its credentials. The client confirms.
Exit conditions	The client is correctly registered to the system.
Exceptions	An exception can be throw if the client inserts some not valid or incorrect data. In that case, the system re-asks the client the data.

NAME	INSERT CURRENT POSITION
Actors	Registered Client
Entry conditions	The client must be successfully login. The client must be in the section for reserving a car. The client must have his mobile phone GPS active.
Flow of events	The client clicks on insert position. The client clicks on insert current position.

Exit conditions	The system, after getting the position of the client, finds and shows the electric cars which state is available next to him.
Exceptions	An exception can be caused if the GPS doesn't work properly or if he detects a position outside the available area. If this happens the system shows an error message.

NAME	**INSERT SPECIFIC POSITION**
Actors	Registered Client
Entry conditions	The client must be successfully login. The client must be in the section for reserving a car.
Flow of events	The client clicks on insert position. The client writes the address in which he wants to take a car.
Exit conditions	The system, after getting the position inserted by the client, finds and shows the electric cars which state is available next to the address chosen.
Exceptions	If the user inserts an address not existing or out from the available area the system throws an exception and shows an error message.

NAME	SELECT A CAR
Actors	Registered Client
Entry conditions	The client must be successfully login. The client must be in the section for reserving a car. The client must have inserted a position in which he wants to take a car.
Flow of events	The client selects a car from the available list clicking on the button next to the car position. The client selects "reserve" option.
Exit conditions	Car successfully reserved. The system changes car and client state and instantiates a relation of reservation between them.
Exceptions	If another client reserves the same car before the process is completed the system throws an exception and show an error message.

NAME	ENABLE SAVING OPTION
Actors	Registered Client
Entry conditions	Client must be registered and successfully login. Client state must be 'on course'.

Flow of events	Client enable saving option from mobile application. Client insert his destination address.
Exit conditions	The system, after getting the position inserted by the client, finds and shows the nearest available special parking area to the destination.
Exceptions	If the user inserts an address not existing or out from the available area, the system throws an exception and shows an error message. If there aren't available special parking area close to the destination the system communicates it to the client

NAME	MONITOR MONEY
Actors	Registered Client
Entry conditions	Client must be successfully login. Client state must be 'on course'.
Flow of events	Client select the monitor money option on the display of the car.
Exit conditions	The system, that is able to calculate real-time the current price of the course, show it to the client on car display, continuing to update it.
Exceptions	No exceptions can be thrown in this situation.

NAME	OPEN THE CAR
Actors	Registered Client
Entry conditions	Client must be successfully login. Client must have reserved a car in the last hour or must be in break. Client must be near the car.
Flow of events	Client select open car option on mobile application. Client scan the car QR code through the application.
Exit conditions	The system unlocks the car in order to let client get in. The system stops reservation countdown and start courtesy countdown.
Exceptions	If camera special interface is not available, the system throws an exception. If QR code is not valid or doesn’t match with the reserved car, the system shows to the client the right position of client car.

NAME	LEAVE THE CAR IN BREAK
Actors	Registered Client
Entry conditions	Client must be successfully login. Client state must be ‘on course’. System has just detected that no one is on the car, the engine is off, all car doors are closed and car is in a safe area.

Flow of events	Client select break option on mobile application.
Exit conditions	Client and car state have been set on break and the car has been lock by the system. Client is still paying.
Exceptions	If client opens again the car before selecting break option, the system denies the user to select this option.

NAME	CLOSE THE CAR AND PAY
Actors	Registered Client
Entry conditions	Client must be successfully login. Client state must be ‘on course’. System has just detected that no one is on the car, the engine is off, all car doors are closed and car is in a safe area.
Flow of events	Client select end of voyage option on mobile application.
Exit conditions	Car has been lock and has been made again available. Client state has been set to ‘dismounted’. Payment transaction took place with the application of special discounts.
Exceptions	If client reopens the car before selecting end of voyage option, the system denies the user to select this option. If car is with more than 80% of battery empty, or client announced a fault via green number, car state is set to ‘not available’.

NAME	SEE AVAILABLE CARS
Actors	Registered Client
Entry conditions	Client must be successfully login. Client state must be in ‘dismounted’ state.
Flow of events	The client clicks on find car button.

Exit conditions	The system sends the available car to the client’s smartphone, that shows them in the map
Exceptions	No Exception can be thrown in this situation

NAME	MODIFY PROFILE
Actors	Registered Client
Entry conditions	Client must be successfully login.
Flow of events	The client goes on his account page. The client clicks on “modify profile”. The client must insert the password. The client changes his credentials, his payment information or his password The client clicks on “save and exit”.
Exit conditions	The client successfully modified his personal information. The system update database with new information.
Exceptions	Exceptions can be thrown if the client doesn’t insert the password correctly. Other error can occur if he decides to change his name and he chooses one already existing. He is not admitted to insert false payment information. If these situations happen the system show an error message to the client and his profile isn’t modified.

NAME	REGISTER AN ASSISTANT
Actors	Employee
Entry conditions	The employee has downloaded and installed PowerEnjoy application. The employee has all data requested for the inscription.
Flow of events	The employee inserts its credentials. The employee confirms.
Exit conditions	The employee is successfully registered in the system as an assistant.
Exceptions	An exception can be throw if the employee inserts some not valid or incorrect data. So the system re-asks the employee to insert data.

NAME	SEE CAR STATE
Actors	Assistant
Entry conditions	Assistant must be successfully login.
Flow of events	The assistant clicks on the “find car” button.
Exit conditions	The system sends all the car at the assistant’s smartphone, that shows them in the map with a different symbol depending of their state. If a car is not available system also shows the entity of the damage.

Exceptions	No Exception can be thrown in this situation

NAME	CHANGE CAR STATE
Actors	Assistant
Entry conditions	The assistant must be successfully logged in.
Flow of events	The assistant selects a car in the map of all cars. The assistant changes the car state clicking the correspondent button to his choice. If he wants to set a car ‘non available’ he also inserts the entity of the damage.
Exit conditions	The state of the car selected is changed
Exceptions	No exception can be thrown in this situation

2.3 Diagrams description

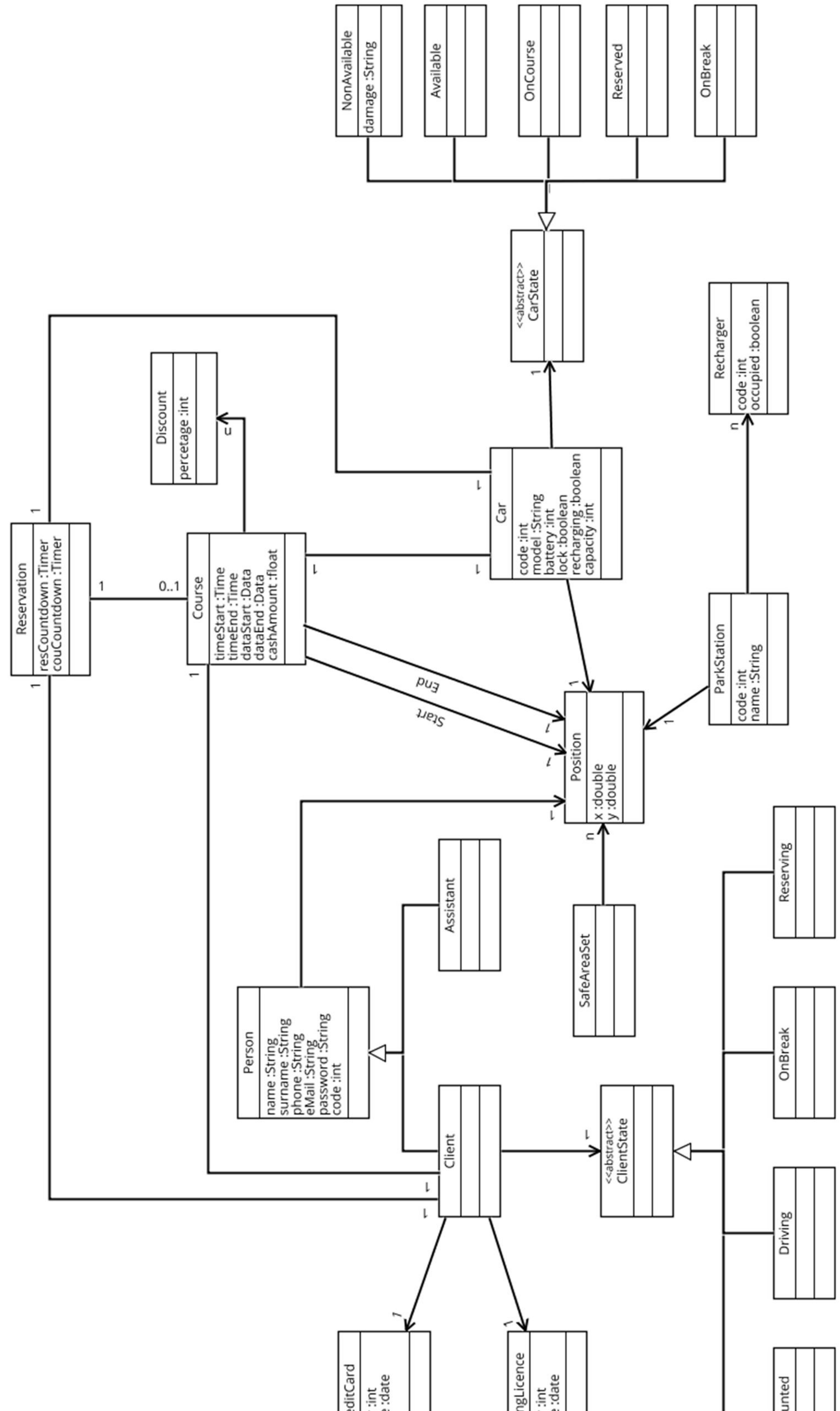
In the Class Diagram we described the objects of our system and the relationships between them. We focused on representing the general structure of our world without analysing the operations and technical details of the objects considered.

With the Sequence Diagrams we decided to describe the temporal relationships between the objects of our system in some important circumstances. The situations analysed are the following: 1) a client who wants to register; 2) a client who wants to reserve a car; 3) a client who wants to modify his profile; 4) a client who is using a car.

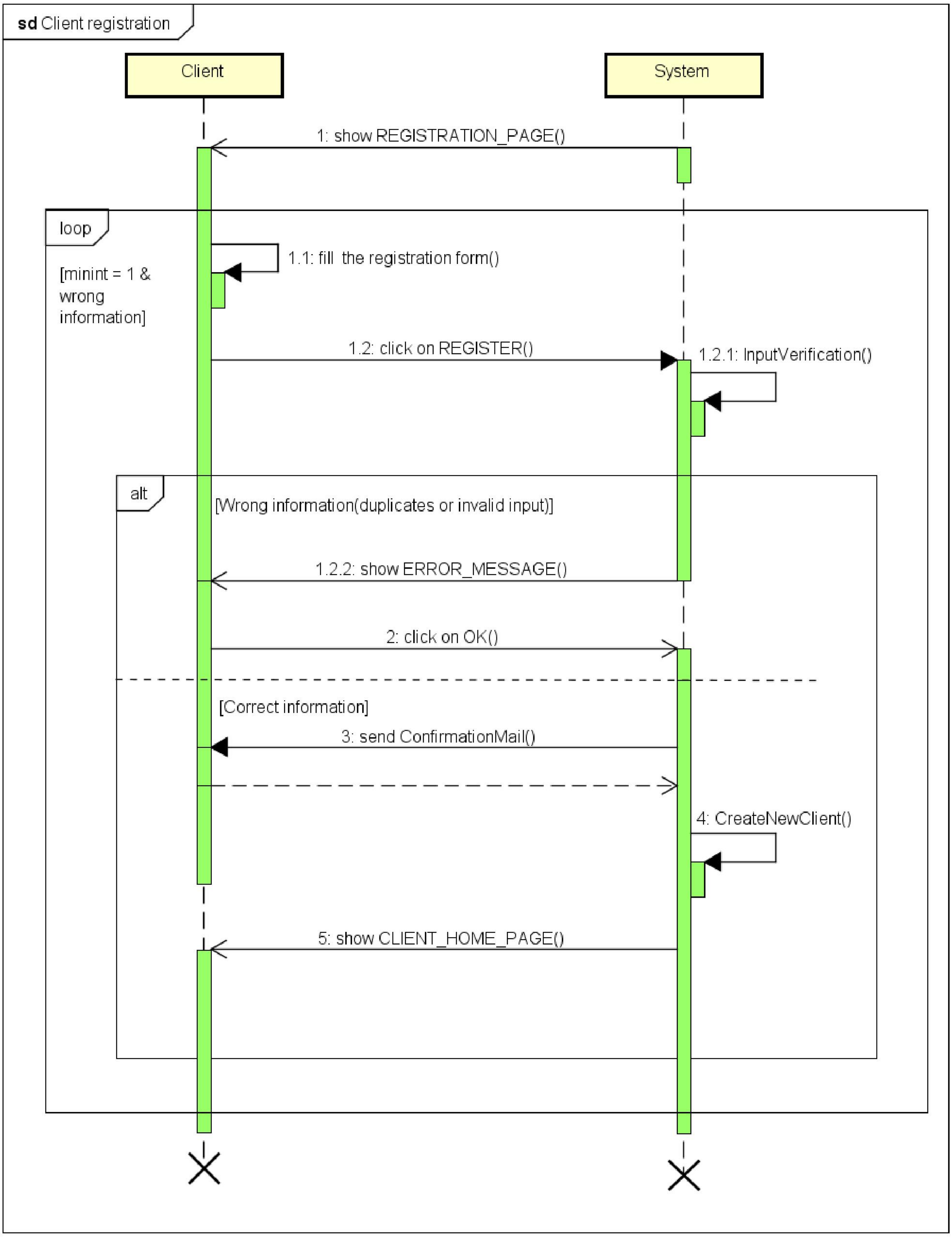
With the Business Process Model and Notation we have represented the flow of actions for the main activity in our system: the usage of a car. the BPMN evidences the different roles of the three parts of the system (the central, the client's app and the car system) and how they interact. In the BPMN we see well all the controls and conditions for each phase of the activity.

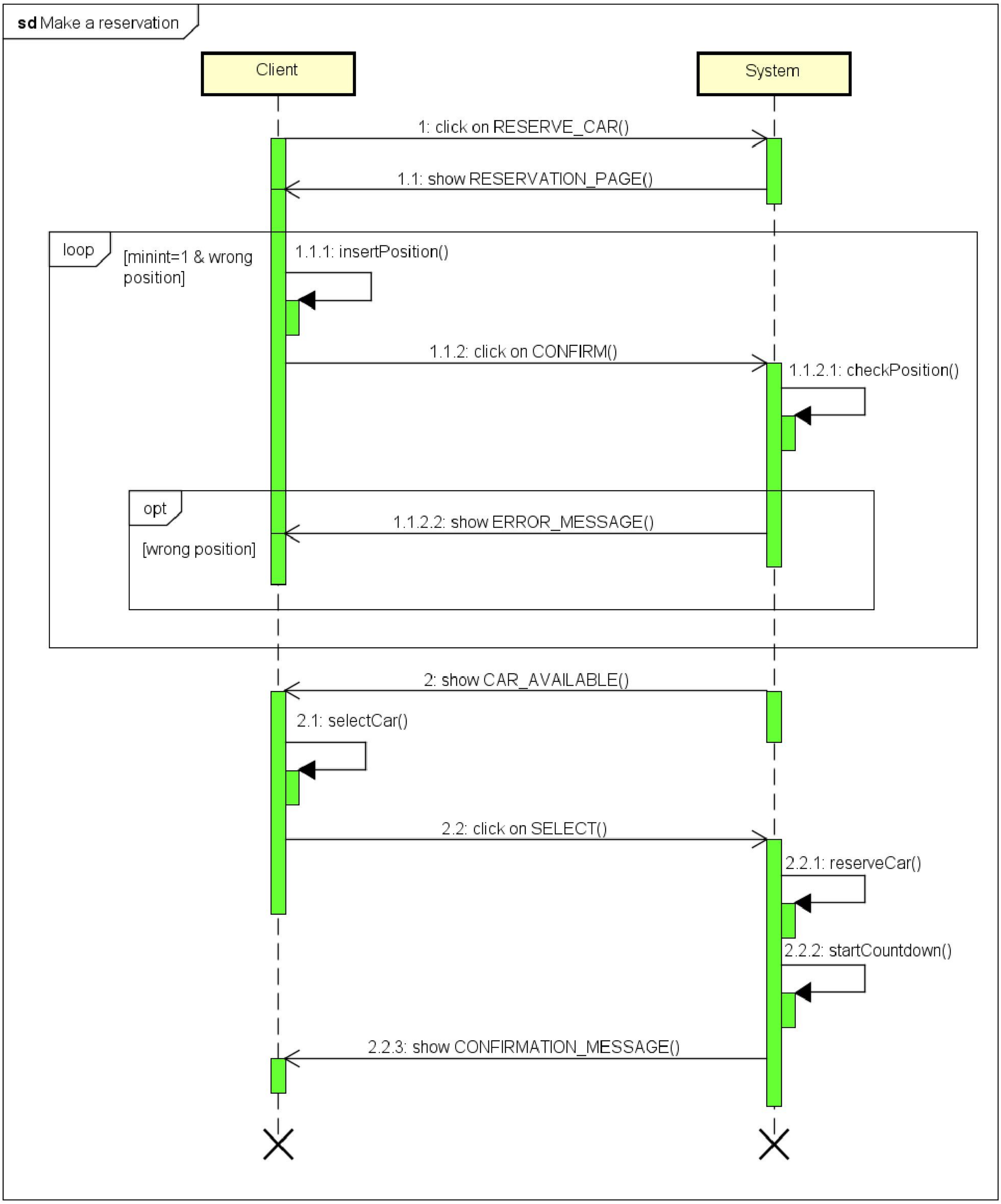
The two State Diagrams concern all possible transactions between existing car and client states. A correct assignment of the states is fundamental for the overall consistency of the world generated by PowerEnjoy system.

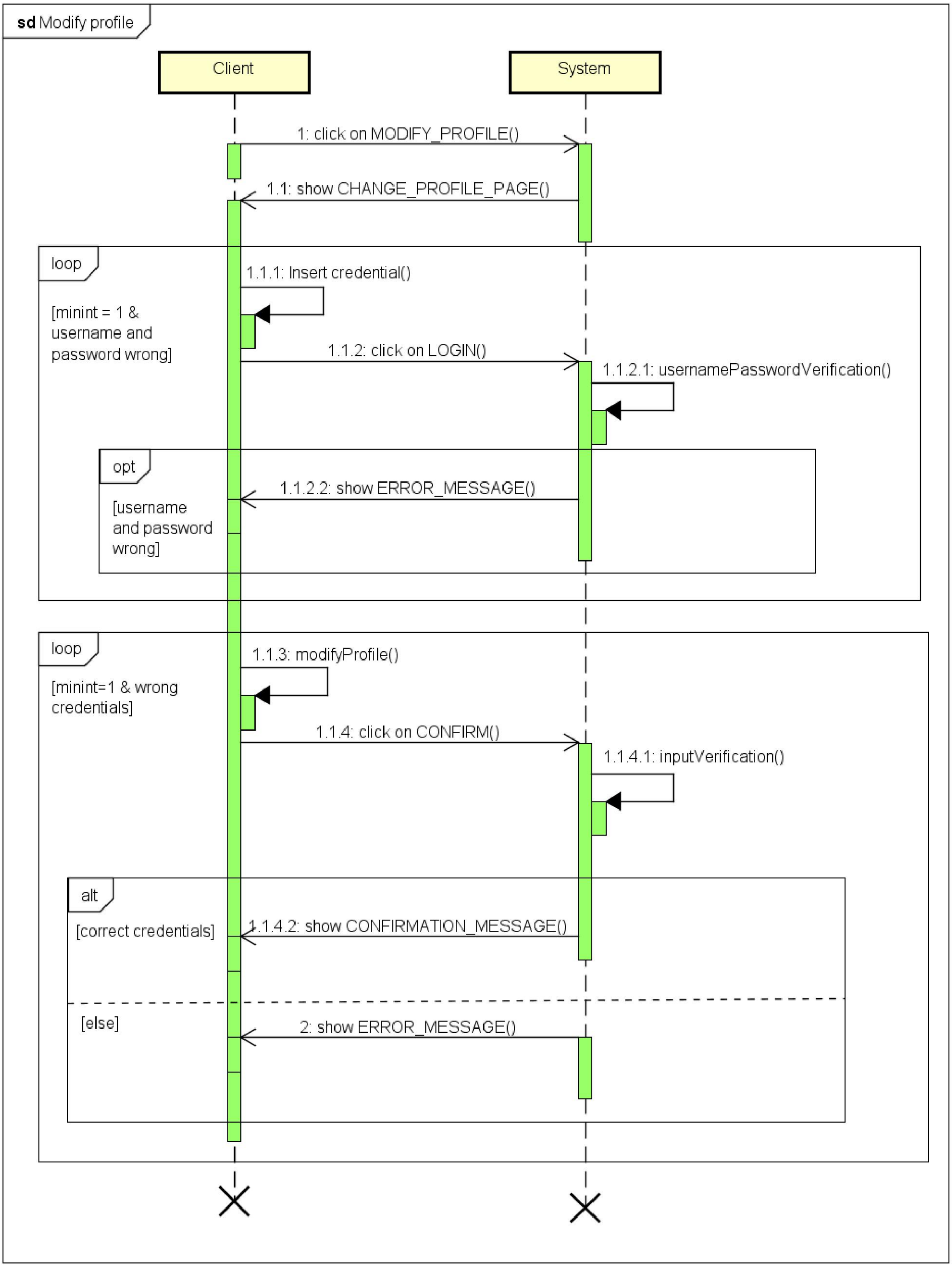
2.4 CLASS DIAGRAM

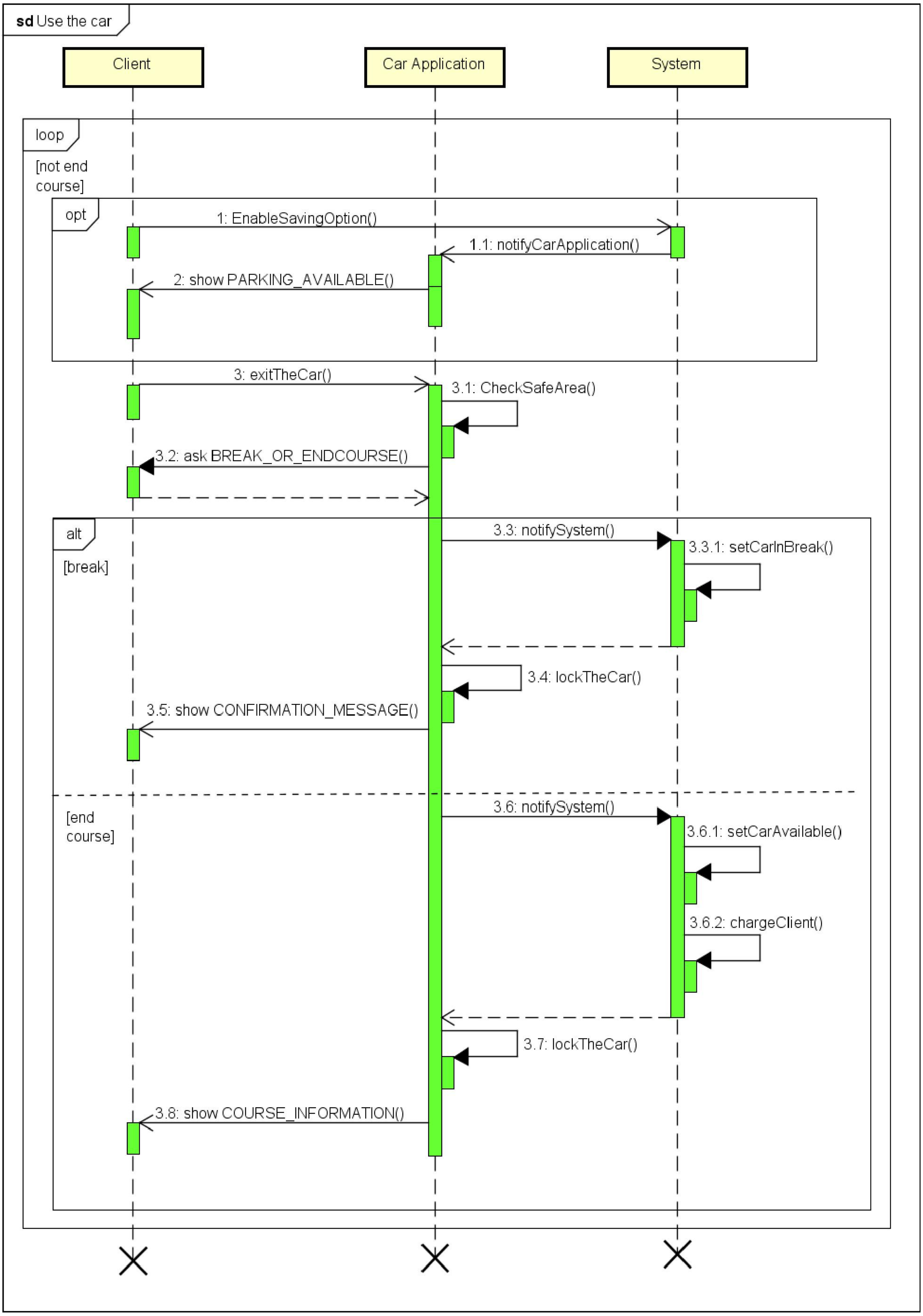


2.5 SEQUENCE DIAGRAMS

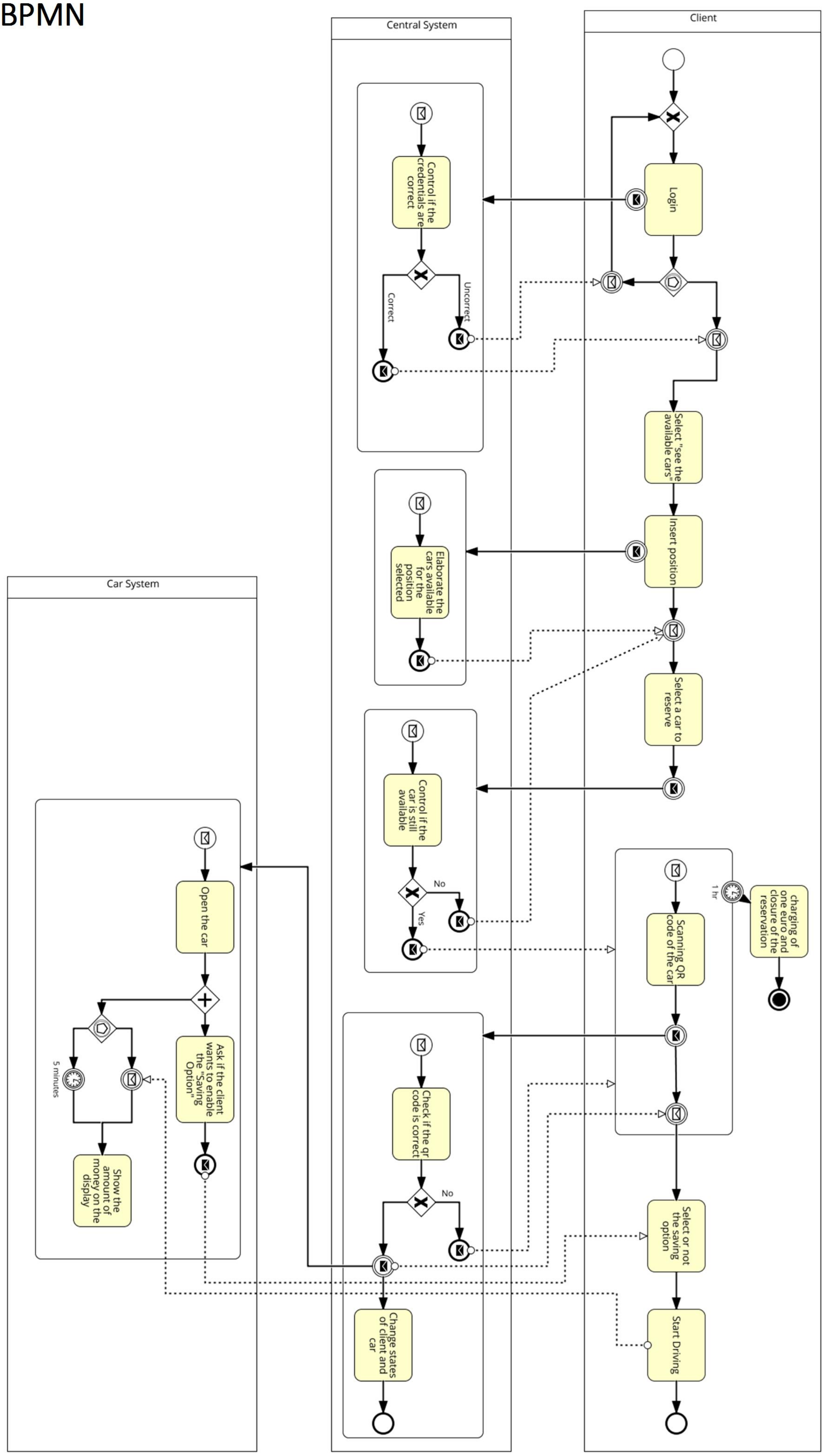

sd Client registration
Client
System
1: show REGISTRATION_PAGE()
loop
[minint = 1 & wrong information]
1.1: fill the registration form()
1.2: click on REGISTER()
1.2.1: InputVerification()
alt
[Wrong information(duplicates or invalid input)]
1.2.2: show ERROR_MESSAGE()
2: click on OK()
[Correct information]
3: send ConfirmationMail()
4: CreateNewClient()
5: show CLIENT_HOME_PAGE()


sd Make a reservation
Client
System
1: click on RESERVE_CAR()
1.1: show RESERVATION_PAGE()
loop
[minint=1 & wrong position]
1.1.1: insertPosition()
1.1.2: click on CONFIRM()
1.1.2.1: checkPosition()
opt
[wrong position]
1.1.2.2: show ERROR_MESSAGE()
2: show CAR_AVAILABLE()
2.1: selectCar()
2.2: click on SELECT()
2.2.1: reserveCar()
2.2.2: startCountdown()
2.2.3: show CONFIRMATION_MESSAGE()

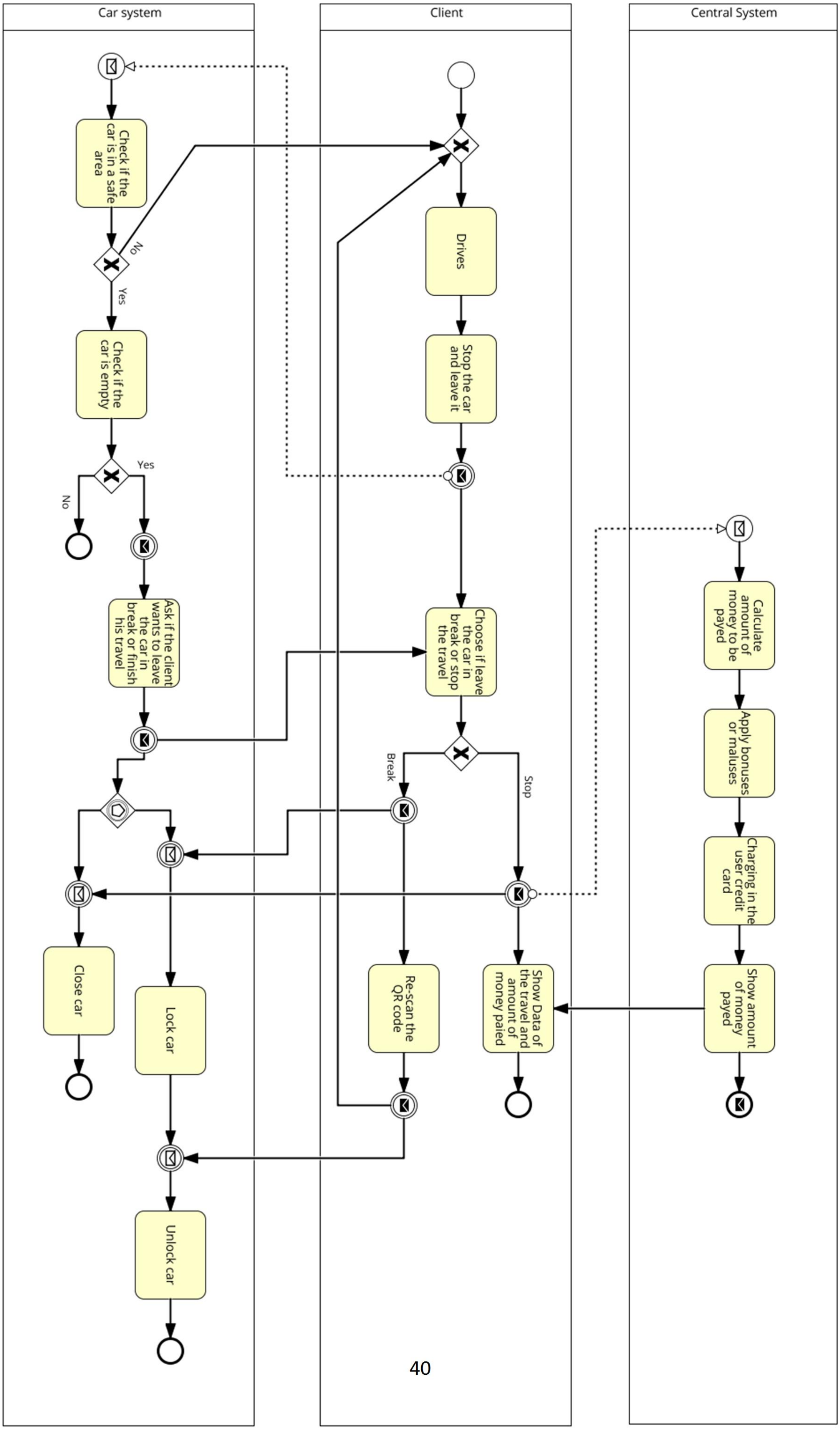

sd Modify profile
Client
System
1: click on MODIFY_PROFILE()
1.1: show CHANGE_PROFILE_PAGE()
loop
[minint = 1 & username and password wrong]
1.1.1: Insert credential()
1.1.2: click on LOGIN()
1.1.2.1: usernamePasswordVerification()
opt
[username and password wrong]
1.1.2.2: show ERROR_MESSAGE()
loop
[minint=1 & wrong credentials]
1.1.3: modifyProfile()
1.1.4: click on CONFIRM()
1.1.4.1: inputVerification()
alt
[correct credentials]
1.1.4.2: show CONFIRMATION_MESSAGE()
[else]
2: show ERROR_MESSAGE()


sd Use the car
Client
Car Application
System
loop
[not end course]
opt
1: EnableSavingOption()
1.1: notifyCarApplication()
2: show PARKING_AVAILABLE()
3: exitTheCar()
3.1: CheckSafeArea()
3.2: ask BREAK_OR_ENDCOURSE()
alt
[break]
3.3: notifySystem()
3.3.1: setCarInBreak()
3.4: lockTheCar()
3.5: show CONFIRMATION_MESSAGE()
[end course]
3.6: notifySystem()
3.6.1: setCarAvailable()
3.6.2: chargeClient()
3.7: lockTheCar()
3.8: show COURSE_INFORMATION()

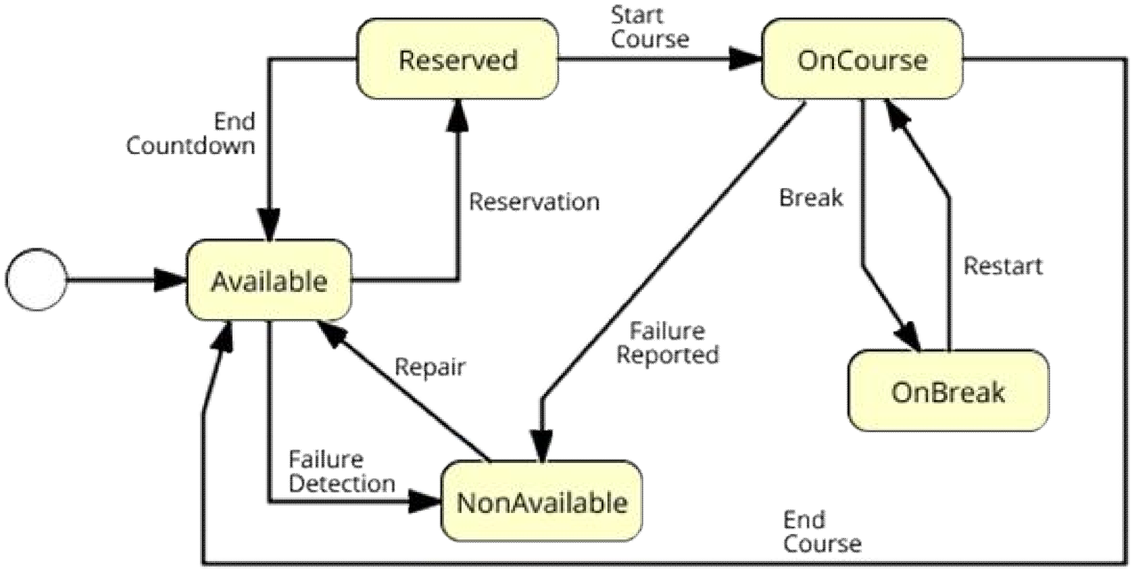
2.6 BPMN

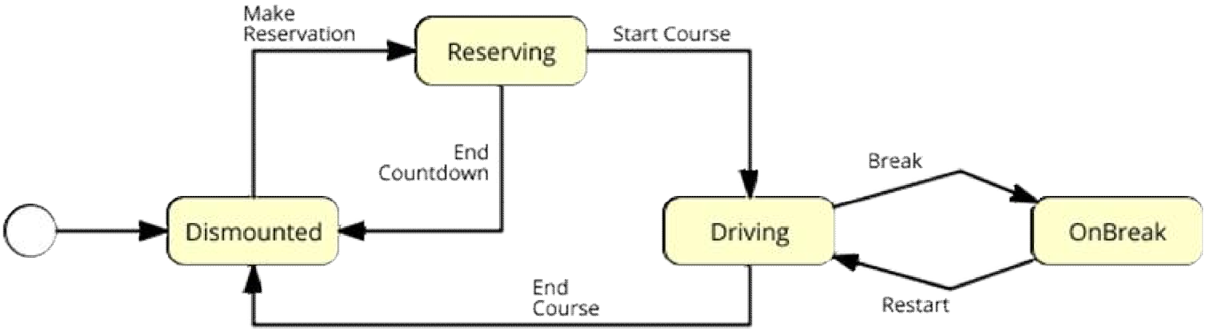


Car System
Central System
Client
Login
Control if the credentials are correct
Correct
Uncorrect
Select "see the available cars"
Insert position
Elaborate the cars available for the position selected
Select a car to reserve
Control if the car is still available
No
Yes
Scanning QR code of the car
1 hr
charging of one euro and closure of the reservation
Open the car
Ask if the client wants to enable the "Saving Option"
5 minutes
Show the amount of money on the display
Check if the qr code is correct
No
Change states of client and car
Select or not the saving option
Start Driving


Car system
Client
Central System
Check if the car is in a safe area
No
Yes
Check if the car is empty
Yes
No
Ask if the client wants to leave the car in break or finish his travel
Close car
Lock car
Unlock car
Drives
Stop the car and leave it
Choose if leave the car in break or stop the travel
Break
Stop
Re-scan the QR code
Show Data of the travel and amount of money paied
Calculate amount of money to be payed
Apply bonuses or maluses
Charging in the user credit card
Show amount of money payed

2.7 STATE DIAGRAMS





3 ALLOY MODELING

3.1 MODEL

In the Alloy we decided to reduce parameters and data showed in order to obtain a more clear and readable model.

We have kept the objects and parameters that participate in the constraints of our system, in particular in the usage of a car in a specific moment.

In the next paragraph, we reported the alloy code, its schema and some representations of the state of the system relative to a car reservation and to a guidance of a client.

3.2 CODE

```
open util/boolean

//----------------------------------------------SIGNATURES----------------------------------------------//

abstract sig ClientState{}

sig DismountedClient extends ClientState{}

sig DrivingClient extends ClientState{}

sig OnBreakClient extends ClientState{}

sig ReservingClient extends ClientState{}

abstract sig Battery{}

sig LowBattery extends Battery{} //minore del 20%

sig HighBattery extends Battery{}

abstract sig Person{
code: Int,
position: lone Position,
} {
code>0
}

sig Client extends Person{
state: one ClientState,
currentReservation: lone Reservation,
currentCourse: lone Course
}

sig Assistant extends Person{}

sig Position
{
x: Int,
y: Int,
} {
x>0
y>0
}

abstract sig CarState {}

sig OnBreakCar extends CarState{}

sig AvailableCar extends CarState{}

sig NonAvailableCar extends CarState{}

sig OnCourseCar extends CarState{}

sig ReservedCar extends CarState{}

sig Car {
code: Int,
battery: Battery,
lock: Bool,
position: one Position,
state: one CarState,
reservation: lone Reservation,
course: lone Course,
} {
code>0
}

sig Course {
client: one Client,
car: one Car,
startPosition: one Position,
}
```

```
sig Reservation{
client: one Client,
car: one Car,
}

sig ParkStation {
code: Int,
position: one Position,
rechargers: set Recharger,
} {
#rechargers >0
code>=0}

sig Recharger {
code: Int,
occupied: Bool,}{
code>=0
}

one sig SafeAreas {
areas: set Position,
}

//---------------------------------------------------------------FACTS-----------------------------------------------
------------//

//the person code are unique
fact personCodeUnique {
all p1, p2: Person | (p1 != p2) implies p1.code != p2.code
}

//positions are unique
fact uniquePosition {
all p1,p2: Position | p1 != p2 implies p1.x != p2.x or p1.y != p2.y
}

//there are no multiple reservations for a client
fact noTwoReservationWithSameClient{
all r1, r2: Reservation | (r1 != r2) implies r1.client != r2.client
}
```

```
//there are no multiple  reservations for a car
fact noTwoReservationWithSameCar{
all r1, r2: Reservation | (r1 != r2) implies r1.car != r2.car
}

//there is a unique driver for each course
fact noTwoDriverForCourse{
all c1, c2: Course | c1 != c2 implies (c1.client != c2.client)
}

//there is a unique car for each course
fact noTwoCarForCourse{
all c1, c2: Course | c1 != c2 implies (c1.car != c2.car)
}

//there are no car with same position
fact noTwoCarWithSamePosition {
all c1,c2: Car | (c1 != c2) implies c1.position != c2.position
}

//relation between client and course
fact clientCourseDoubleRelation {
all cl: Client, co: Course |  cl.currentCourse = co iff co.client = cl
}

//relation between car and course
fact carCourseDoubleRelation {
all ca: Car, co: Course |  ca.course=co iff co.car=ca
}

//relation between client and reservation
fact clientReservationDoubleRelation {
all cl: Client, r: Reservation | cl.currentReservation=r iff r.client=cl
}

//relation between car and reservation
fact carReservationDoubleRelation {
all ca: Car, r: Reservation | ca.reservation=r iff r.car=ca
}

//a client is in Reserving state if and only if he has a reservation and no course
fact consistencyOfClientReservationState {
all c: Client | c.state=ReservingClient iff (#c.currentReservation=1 and #c.currentCourse=0)
}
```

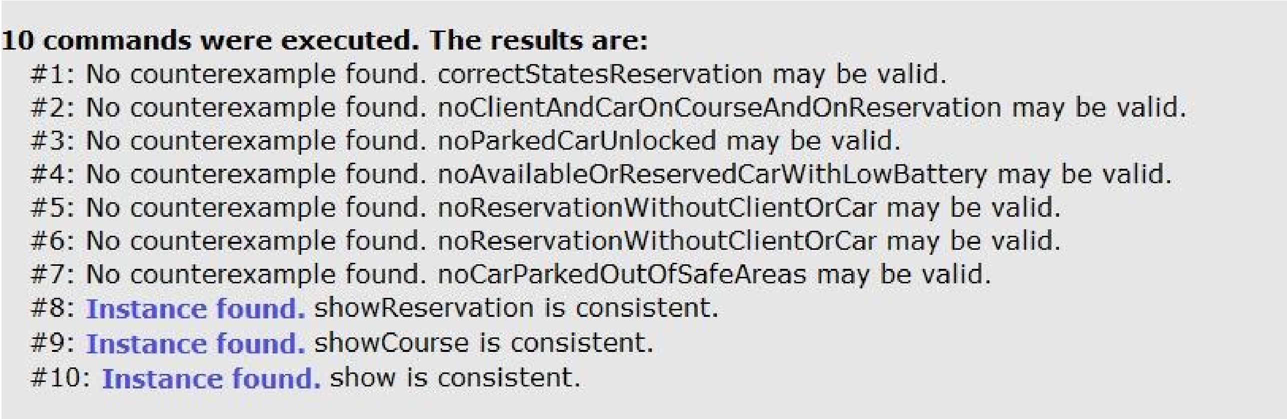
```
//a car is in Reserved state if and only if it has a reservation and no course
fact consistencyOfCarReservationState {
all c: Car | c.state=ReservedCar iff (#c.reservation=1 and #c.course=0)
}

//a client is in Driving or OnBreak state if and only if he has a course and no reservation
fact consistencyOfClientCourseState {
all c: Client | (c.state=DrivingClient or c.state=OnBreakClient) iff (#c.currentReservation=0
and #c.currentCourse=1)
}

//a car is in OnCourse or OnBreak state if and only if he has a course and no reservation
fact consistencyOfCarCourseState {
all c: Car | (c.state=OnCourseCar or c.state=OnBreakCar) iff (#c.reservation=0 and
#c.course=1)
}

//for all courses the cars and clients associated are OnCourse/Driving or OnBreak states
fact stateOfClientAndCarInCourse{
all c: Course |  (c.client.state = DrivingClient and c.car.state = OnCourseCar) or
                                        (c.client.state = OnBreakClient and  c.car.state =
OnBreakCar)
}

//car not on course are locked
fact carNotOnCourseLocked {
all c: Car | c.state = OnCourseCar iff c.lock = False
}

//battery discharged => non available
fact  carDischargeNonAvailable {
all c: Car | c.battery=LowBattery
                          implies
                          ( c.state = NonAvailableCar or c.state = OnCourseCar or c.state =
OnBreakCar)
}

//reserved/available car => battery not discharged
fact carReservedAndAvailableWithBatteryCharged{
all c: Car | (c.state = ReservedCar or c.state = AvailableCar)
                          implies
                          c.battery = HighBattery
}
```

```
//there aren't park stations with same position
fact noTwoParkStationWithSamePosition {
all p1,p2: ParkStation | p1 != p2 implies p1.position != p2.position
}

//park station code are unique
fact parkStationCodeUnique {
all p1,p2: ParkStation | p1 != p2 implies p1.code != p2.code
}

//recharger code unique in each park station
fact rechargerCodeUniqueInParkStation {
all r1, r2: Recharger, p: ParkStation | ( r1 in p.rechargers and r2 in p.rechargers ) implies
r1.code = r2.code
}

//there aren't two park station with the same recharger
fact noParkStationWithSameRecharger {
all p1, p2: ParkStation, r: Recharger | ( p1 != p2 ) implies !(r in p1.rechargers and r in
p2.rechargers)
}

//all car in available or non available or on break or reserved state are parked in a safe area
fact carAvailableAndOnBreakInSafeArea {
all c: Car | some  s: SafeAreas |
                                  (c.state = AvailableCar or c.state = NonAvailableCar or c.state =
OnBreakCar or c.state = ReservedCar)
                                  implies
                                  c.position in s.areas
}

//there are only battery related to cars
fact batteryInACar
{
all b: Battery | one c:Car | c.battery=b
}

//there are only carState related to cars
fact onlyStatesReletedToCar
{
all cs:CarState | some c:Car | c.state=cs
}
```

```
//there are only clientState related to client
fact onlyStatesReletedToClient
{
all cs:ClientState | some c:Client | c.state=cs
}

//there are only recharger related to park station
fact onlyRechargerRelatedToParkStation
{
all r: Recharger | one p: ParkStation | r in p.rechargers
}

//-------------------------------------------------------PREDICATES----------------------------------------
-----------//

pred reserveACar[cl:Client, ca:Car, r:Reservation ]{
 r in cl.currentReservation and ca in r.car
}

pred inDriving[cl:Client, ca:Car, co:Course]{
cl in co.client and ca in co.car
}

//predicate that shows a world that underlines course properties
pred showCourse
{
	#Reservation=0
	#ParkStation=1
	#Course=1
	#Car=1
	#Client=1
}

//predicate that shows a world that underlines reservation properties
pred showReservation
{
	#Reservation=1
	#ParkStation=1
	#Course=0
	#Car=1
	#Client=1
	#Recharger=1
}
pred show{}
```

```
//-------------------------------------------------ASSERTION------------------------------------------------//

//check if there are setted correctly the states of the client and the car in a reservation
assert correctStatesReservation
{
        all cl:Client, ca:Car, r: Reservation| reserveACar[cl, ca, r] implies
(cl.state=ReservingClient and ca.state=ReservedCar)
}

check correctStatesReservation

//check that doesn't exsist cars or clients both in course and in reservation
assert noClientAndCarOnCourseAndOnReservation
{
        all cl:Client, ca:Car, co:Course, r:Reservation| inDriving[cl, ca, co] implies (r.client!=cl
and r.car!=ca)
}

check noClientAndCarOnCourseAndOnReservation

//check that there aren't car parked (and so in available or non-available or onbreak or
reserved state) and unlocked
assert noParkedCarUnlocked
{
        no c: Car| c.lock = False and
        (c.state = AvailableCar or c.state = NonAvailableCar or c.state = OnBreakCar or c.state
= ReservedCar)
}

check noParkedCarUnlocked

//check that there are no car reserved or available with low battery
assert noAvailableOrReservedCarWithLowBattery
{
        no c: Car |     (c.state = AvailableCar or c.state = ReservedCar) and c.battery =
LowBattery
}

//check that there are no reservation without car or client associated
assert noReservationWithoutClientOrCar
{
        no r: Reservation | #r.client=0 or #r.car=0
}
```

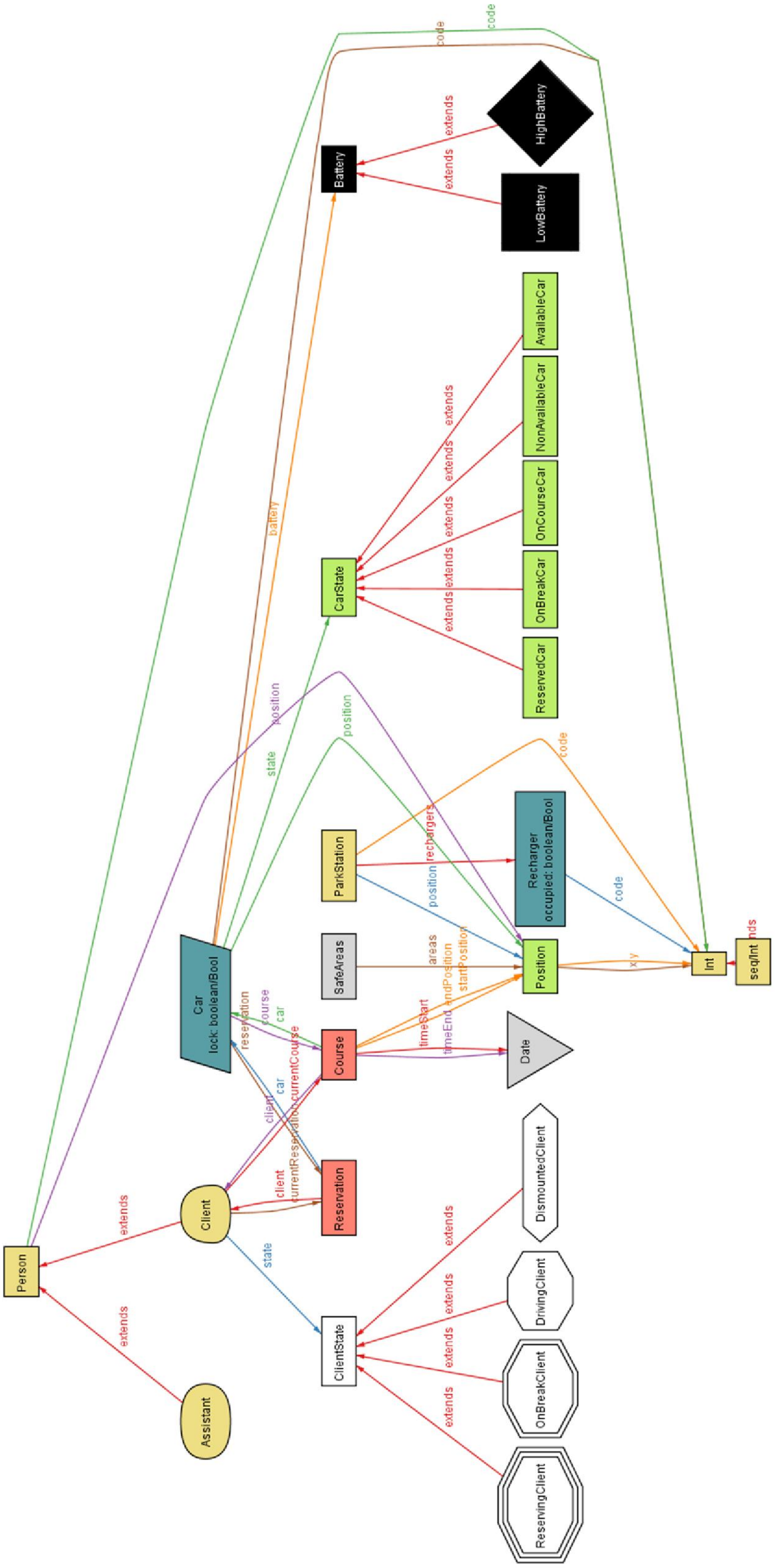
```
check noReservationWithoutClientOrCar

//check that there are no course without car or client associated
assert noCourseWithoutClientOrCar
{
        no c: Course| #c.client=0 or #c.car=0
}

check noReservationWithoutClientOrCar

//check that no car parked aren't in a safe area
assert noCarParkedOutOfSafeAreas
{
        no c: Car | all s: SafeAreas | c.position not in s.areas and (c.state = AvailableCar or
c.state = NonAvailableCar or c.state = OnBreakCar or c.state = ReservedCar)
}

check noCarParkedOutOfSafeAreas

//----------------------------------------------RUN--------------------------------------------------------------//

run showReservation for 2

run showCourse for 2

run show for 2
```

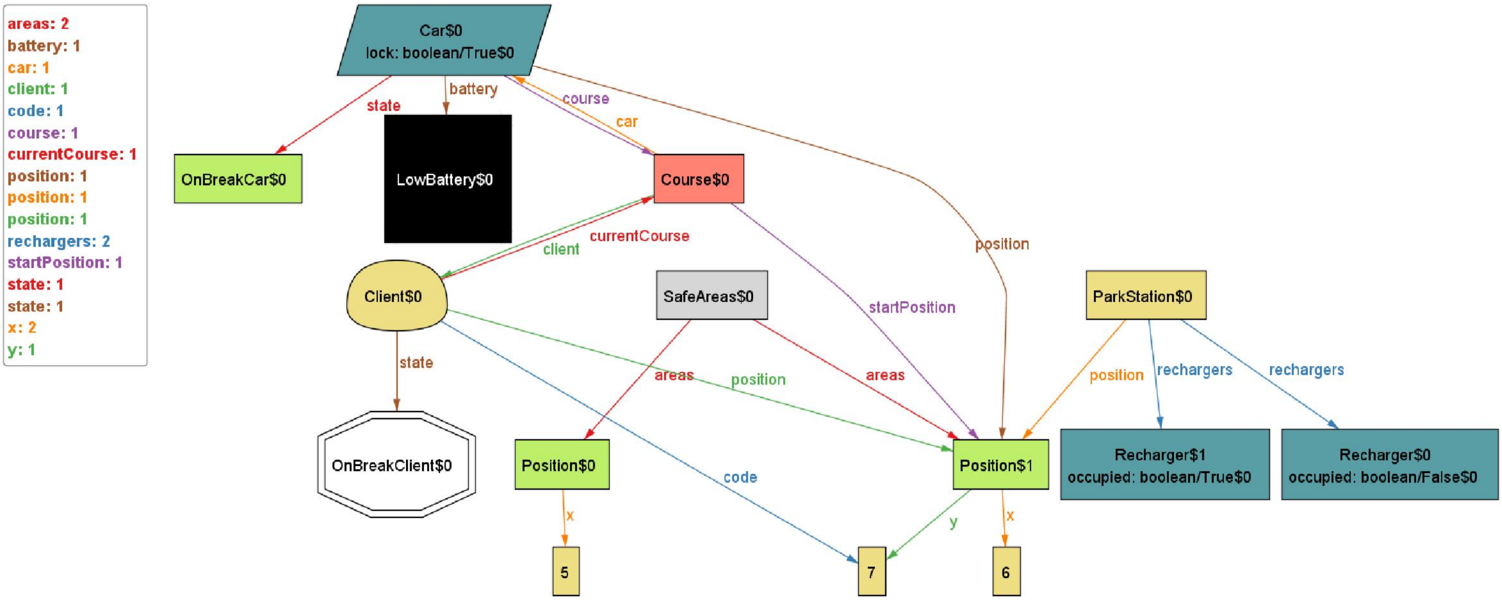


```
10 commands were executed. The results are:
   #1: No counterexample found. correctStatesReservation may be valid.
   #2: No counterexample found. noClientAndCarOnCourseAndOnReservation may be valid.
   #3: No counterexample found. noParkedCarUnlocked may be valid.
   #4: No counterexample found. noAvailableOrReservedCarWithLowBattery may be valid.
   #5: No counterexample found. noReservationWithoutClientOrCar may be valid.
   #6: No counterexample found. noReservationWithoutClientOrCar may be valid.
   #7: No counterexample found. noCarParkedOutOfSafeAreas may be valid.
   #8: Instance found. showReservation is consistent.
   #9: Instance found. showCourse is consistent.
   #10: Instance found. show is consistent.
```

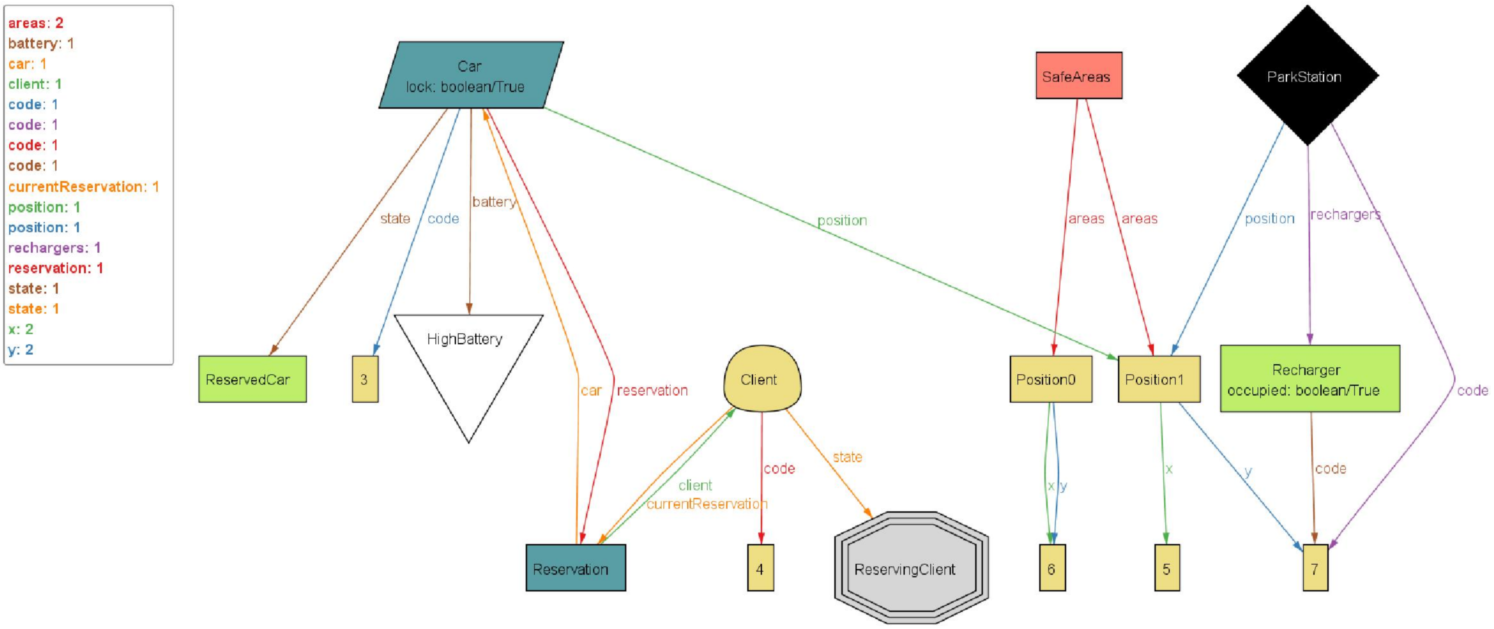
3.3 WORLD GENERATED

This is the metamodel of the world generated by alloy in which we can see the relationship between all signatures

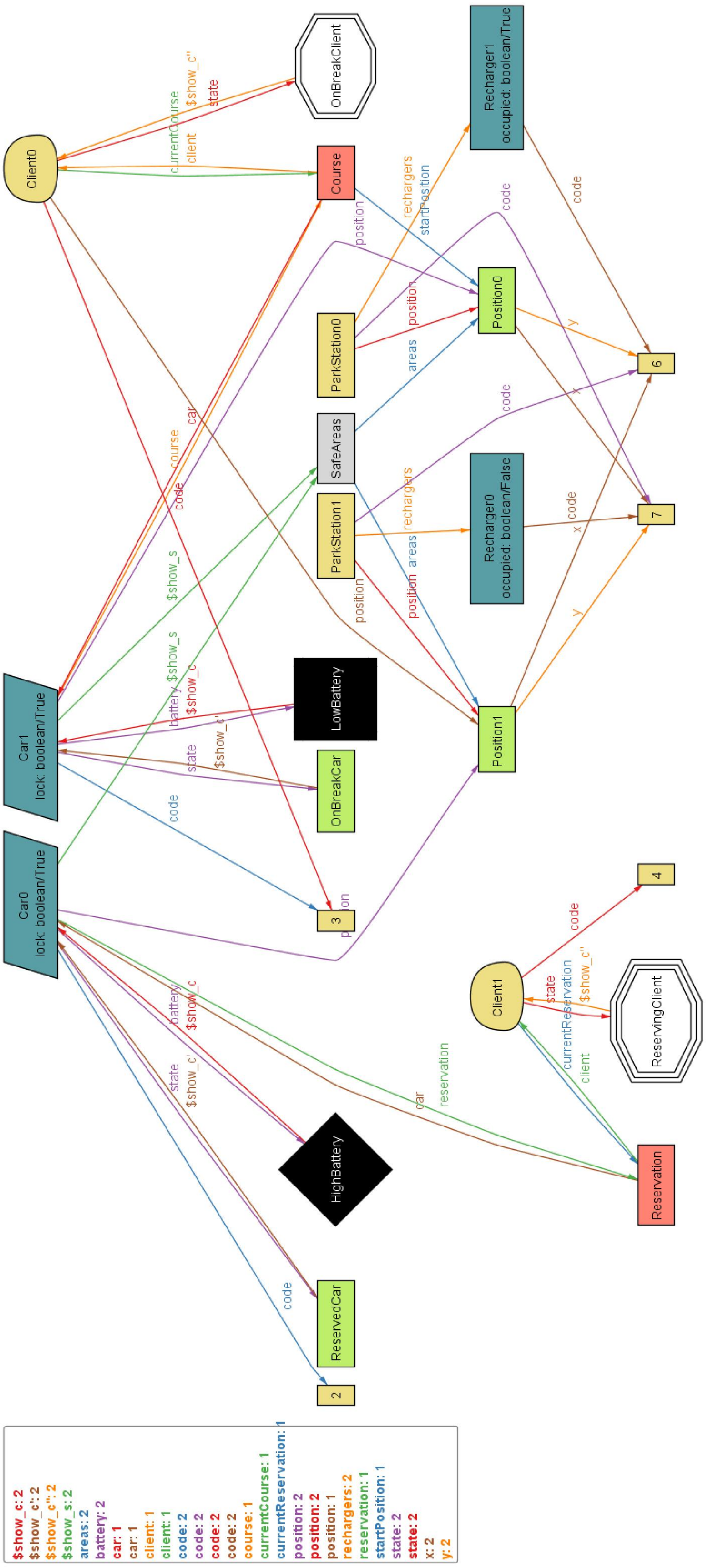




This model underlines the relationship between Course and Car/Client. In particular we have a car who left the car "On Break" with Low Battery in a Safe Area in which there are two Rechargers.



This model underlines the relationships between Reservation and Client/Car. In particular there is a Client who reserved a Car. We can see that he is in "ReservingClient" state. Furthermore, the car has high battery level and is parked in a safe area.



This last picture shows a more generic model in which there are both a course and a reservation.

All clients have exactly one reservation or one course and all state are consistent

4 OTHER INFORMATION

4.1 USED TOOLS

The tools we used to create this document are:

- Microsoft Word 2016 : to write and assemble the document
- Dropbox e GitHub : to share work
- Signavio Academic : for BPMN and Use Case, Class and State diagrams
- Pencil : to draw mockup
- Astah : for Sequence diagrams
- Alloy Analizer 4.2 : to prove the consistency of our model

4.2 HOURS OF WORK

The writing of this document took overall about 25 hours of head-work.

The amount of work has been divided equally between the 3 members of the group.

About 10 hours of 25 were spent working together in order to set up correctly and better coordinate each part of the work.

Furthermore, last 2 hours of head-work have been spent for reread, revise and assemble the whole document.

4.3 REFERENCE DOCUMENT

- Specification document : Assignments AA 2016-2017.pdf
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications
- Examples documents :
    - RASD sample from Oct. 20 lecture.pdf (myTaxiService)
    - RASD_example_SWIMv2.pdf
    - RASD_meteocal-example1.pdf
    - RASD_meteocal-example2.pdf