POLITECNICO
MILANO 1863

Requirement Analysis and Specifications Document

Lo Bianco Riccardo - Manzoni Mirco - Mascellaro Giuseppe

December 9, 2016
v1.0

Contents

1 Introduction

1.1 Description of the problem

PowerEnJoy is a car sharing service that exclusively employs electric cars. The company is in need of a system that allows users to find the locations of available cars located in the geographical areas they choose to explore. Then, they may complete the reservation of a car among those found with the research tool. After that, if the user who made the reservation launches the car doors unlock command and PowerEnJoy detects he is nearby, then the car doors are actually unlocked.
During the ride, the driver is notified of the current bill in real-time through the screen. At the end of the ride expenses are automatically charged on the driver's count. Since all the vehicles are electric, it is fundamental to keep them properly charged, so the users will be incentivized in being virtuous through discounts and penalty fees. A saving option will be provided in order to help users make the right choices both to save money and to help the PowerEnJoy maintain its high quality service.

1.2 Goals

PowerEnJoy must provide the following main features:

- **[G1]** Whenever a user needs to rent a car, he can do it for a fi xated price.
- **[G2]** Whenever a user wants to use a car, nobody else can use that car at the same time.
- **[G3]** Let users find and reach cars.
- **[G4]** Let drivers find and reach safe parking areas and locations.
- **[G5]** Encourage the sharing of a single car.
- **[G6]** Keep cars battery at a level such that they can be used for the greatest possible amount of time.
- **[G7]** Keep cars always well distributed on the territory.
- **[G8]** Make sure that most of the cars are ready to use and in a good condition.
- **[G9]** Identify who the driver is for security purposes.

1.3 Domain assumptions

We suppose these assertions to be true in the analyzed domain:

- **[D1]** Personal information provided in the sign up phase are true.
- **[D2]** Cars are always connected to PowerEnJoy VPN (Virtual Private Network).
- **[D3]** Cars' GPS is never neither switched off nor damaged.
- **[D4]** Cars' locations are known by GPS.
- **[D5]** Users who have an open reservation are properly located by GPS through their mobile devices.
- **[D6]** Payments management is delegated to a third party company.
- **[D7]** All sensors report correct information and they are neither switched off nor damaged.
- **[D8]** Every possible technical issue is detected by the sensors.
- **[D9]** Each power plug can be linked to at most one car at the same time.
- **[D10]** Power grid stations always dispense electricity.
- **[D11]** Cars' screens are never switched off nor damaged.
- **[D12]** The number of available slots in a special parking area is always known by PowerEnJoy though its VPN.
- **[D13]** The user who inserts his password on the screen is the same who actually drives the car.

1.4 Glossary

In this paragraph, we go through some recurrent terms that deserve a complete definition to avoid misunderstandings along the discussion:

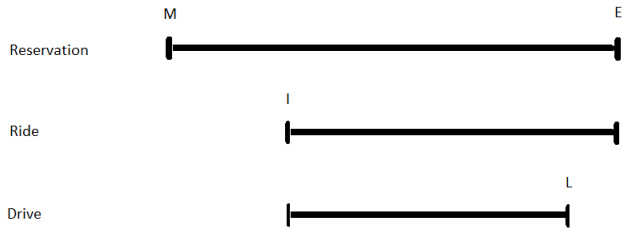
- **Bypass:** hardware component provided to the operators in order to unlock car doors without having any active reservation.
- **Car:** one of the electric vehicles owned by PowerEnJoy.
- **Discount:**
  - **A:** PowerEnJoy detects the user took at least two other passengers onto the car. A 10% discount is applied on the last ride's bill.
  - **B:** the car is left no more than 50% of the battery empty. A 20% discount is applied on the last ride's bill.
  - **C:** the car is left in a special parking area and it is plugged to a power grid station before ending the ride. A 30% discount is applied on the last ride's bill.
- **Driver:** a user who already ignited the engine of the car he rented.

- **Fee:**
  - **A:** a user makes a reservation but he does not unlock the car doors within one hour from the reservation. A 1 € fee is applied.
  - **B:** a car is left more than 3 kms away from the nearest power grid station or with more than 80% of empty battery. A 30% fee is applied on the last ride's bill.
  - **C:** the car is left in a special parking area and it is plugged to a power grid station before ending the ride. A 1 € per minute fee is applied until the user unlocks the car doors without igniting the engine.
- **Guest:** an unlogged visitor of the application. Guests are only allowed to browse the map, but they can not make reservations.
- **Minor issue:** an issue not detectable by sensors, such as dirt into the car cabin or damages to the car seats. This kind of issues can be reported by users through the apposite functionality.
- **Operator:** an employee of PowerEnJoy who is in charge of resolving all kinds of issue in the service.
- **Passenger:** every person who boards a car, driver included.
- **Payment information:**
  - Payment method name
  - Surname and name of the owner
  - Card number
  - Expiration date
  - CVV
- **Personal information:**
  - Surname and name
  - Nationality
  - Date of birth
  - City of birth
  - Username
  - Phone number
  - E-mail address
  - Driving license
- **Power grid station:** the energy turrets where users can leave cars to refill batteries.
- **Power plug:** there is one and only one power plug per power grid station.
- **Reservation:** functionality provided to the users to reserve a single car at one time.

- **Ride:** time gap between the ignition of the engine and the locking of the car doors.
- **Safe parking area:** a parking area included in a list of legal geographical areas to leave a car. Safe parking areas are the only places where a car can be left to put an end to a ride.
- **Screen:** system terminal embedded on every car and connected to the central server. The screen is used to directly interact with the user during a ride.
- **Special parking area:** a subset of safe parking areas where a fixated number of power grid stations is present.
- **Technical issue:** an issue that can be detected by the car's sensors, such as mechanical and electrical problems, battery level and eventual collisions.
- **User:** a guest who provided personal and payment information which have been verified. Users have access to the full set of PowerEnJoy features.

1.5 Specification of some terms

The terms "reservation" and "ride" can be clarified through the following diagram:



Where the labels mean:

- **M**: make a reservation
- **E**: end of the reservation (invoice is generated)
- **I**: ignite engine
- **L**: lock car doors

1.6 Text assumptions

In this section we analyze all the properties of the application domain of the system in order to describe the environment in which it operates.

- A guest can only register, surf the list of available cars on the map and consult the service's rules.
- The A discount can be applied if and only if the number of passengers is greater or equal to three, both at the beginning and at the end of the ride.
- Each special parking area is composed of a fixated number of parking slots.
- Each parking slot in a special parking area is associated to exactly one power grid station.
- Each power grid station has exactly one power plug, so it is possible to determine wether a car is plugged in or not.
- Road fines are discussed with traffic authorities by the company's legal department.
- The validity of the driving license number associated with the user's profile is verified by the driver licensing authority, which is completely external from PowerEnJoy.
- The validity of the payment information associated with the user's profile will have to be verified by the payment company, which is completely external from the PowerEnJoy.
- Once a reservation is made, the user who made it may cancel it.
- Payments are charged at the end of each ride, so it is possible to correctly evaluate the amount of fees and discounts.
- An operator takes care of cars' repositioning periodically, so it is less probable that not uniform distribution issues occurs.
- A driver can not temporary park his car and exit from it. This assumption is made to prevent drivers from leaving cars in parking areas not included in the set of safe parking areas. In a future perfective development this assumption will be erased.
- A discount regarding power grid stations is applied if and only if the car is in a special parking area and the car is actually charging.
- The set of special parking areas is pre-defined in PowerEnJoy.
- Car batteries are considered almost empty when they reach 20% level.
- A user who unlocks the car and does not ignites the engine within 5 minute has to pay the C fee. For all the time the driver does not ignite the engine, he is encouraged to do so or leave the car through the screen.
- Power grid stations are located only in special parking areas.

- A safety lock is present on each power plug on the cars: locking and unlocking car doors means that also the power plug on the car is either locked or unlocked. This is to prevent people from disconnect the cars from their plugs unless they want to make a ride.
- Cars' windows close automatically whenever a ride ends.
- Sensors capable of detecting whether a person is sitting or not on the car seats are present on every car, so it is possible to determine how many passengers are there on each car.
- Discounts and fees are calculated all together at the end of each reservation. Discounts are calculated first in the order A-B-C, then the eventual fee B is considered.
- Map navigation, tow trucks and mechanics services are managed by external affiliated companies.
- Assurances and other necessary documentations are periodically renewed by external affiliated companies.

1.7 Proposed system

The best solution to carry on the project is to develop a web platform, both in the form of a website and of a mobile application. It is fundamental to develop an application that is multi-platform, since one of the wanted features is that the user who reserved a car must be able to ulock it with his phone. Because of this we plan to develop a website available in all the most common browsers (Mozilla Firefox, Google Chrome, Microsoft Edge) and a mobile app compatible with Android, OSX and Windows Phone. A plugin for wearable devices could also serve to this functionality.
The mobile application to be developed interacts with previously installed sensors. The system will retrieve data periodically from the cars' sensors to find technical issues in real time. The website will be available in all the most common browsers (Mozilla Firefox, Google Chrome, Microsoft Edge) and the mobile app will work in Android, OSX, Windows Phone.

1.8 Actors identifying

The actors involved in PowerEnJoy are:

- Guest: a person who can access a limited number of the PowerEnJoy's features, e.g. the research tool. They can neither make reservations nor access to any features that requires the possibility of online payments.
- User: a person who has registered and therefore has provided his personal and payment information.
- Driver: a user who has made a reservation for a car and now is driving it.
- Operator: a person who takes care of cars' maintenance and picks up various kinds issues whenever they happen. We use this term to identify both logged and unlogged operators (loggin in is required also from operators, so we can identify who is active and who is not).

1.9 Stakeholders

The one and only stakeholder for this project is represented by the professor who defined the assignment. The deadline for the submission of the complete document is the end of the current semester. The final submission should provide a clear and complete documentation for the development of the system, along with use cases analysis and tests.
The completeness of the document will be compromised by the need of focusing on the major features of the system. Nonetheless we'll try to maintain a high level of consistency all along the dissertation and to develop as many aspects as possible.
Concerning the target of the application, we can think of the standard user as a stable user: the aim of PowerEnJoy should be to convince the customers to use it more than just once.

1.10 Other considerations concerning the system

1.10.1 Fees and discounts

Fees and discounts are applied at the end of each ride in the order indicated in the assumptions paragraph. However, we can notice that not every kind of discount is compatible with every kind of fee and viceversa. In particular:

1. If fee A is applied on a reservation, no other fee or discount can be applied on the same reservation.
2. If discount C is applied on a ride, fee B can not be applied on the same ride.
3. All other discounts and fees are compatible.

1.10.2 Possible future extensions

The system as described in the document is only the starting point of a more complex deployment. In particular, we assumed that no stops are allowed during a ride, a feature that can be modeled in a future developement. Different kinds of discounts and fees can be introduced as well, based on the analysis of users' behaviour. What we tried to produce in the RASD is the blueprint of a system that is both complete as it is in a first deployment and flexible enough to be updated and easily integrated with new features.

2 Requirements

2.1 Functional requirements

In the following paragraph we describe the main use cases in detail. The list of use cases presented in the following pages does not cover all the possible events that can occurr in the analyzed domain, but it is sufficient to explore all the features of the system, both in normal and exceptional situations. When we talk about pages and buttons it is only to help the reader figuring the ongoing flow of events: pages' structure will be examined in depth in the Design Document.

1. **[G1] Whenever a user needs to rent a car, he can do it for a fixated price.**
   - R1 Sign up functionality.
   - R2 Verification of driving license and payment information.
   - R3 Log in functionality.
   - R4 Prevention of car doors unlocking unless the user is into a 10m range.
   - R5 Start charging expenses for a given amount of money per minute as soon as the engine is ignited.
   - R6 Chargingt of expenses on the user's account after the end of each ride.
   - R7 Consult last rides' invoices.
2. **[G2] Whenever a user wants to use a car, nobody else can use that car at the same time.**
   - R1 Make reservations valid for a single car at the same time.
   - R2 Reservations expires after one hour and the A fee is charged on the user's payment method, then the car is available again.
   - R3 Reserved cars are marked as unavailable.
   - R4 Lock car doors after each ride.
   - R5 Stop counting expenses and generate an invoice whenever a ride terminates.
   - R6 Restore cars' availability after each ride.
   - R7 Cancel a reservation before starting the ride.
3. **[G3] Let users find and reach cars.**
   - R1 Find cars located nearby or by specifying an address.
   - R2 Provide navigation tips to to reach a reserved car.
4. **[G4] Let drivers find and reach safe parking areas and locations.**
   - R1 Guide each driver to the chosen destination area.
   - R2 If the money saving option is enabled, guide the driver to a special parking area according to his final destination.

5. **[G5] Encourage the sharing of a single car.**

   R1 Apply the A discount.

6. **[G6] Keep cars battery at a level such that they can be used for the greatest possible amount of time.**

   R1 Show special parking areas.

   R2 Apply the B discount.

   R3 Apply the C discount.

   R4 Periodically, if there is a car with almost empty battery, send a notification to operators who will intervene.

   R5 Restore cars' availability once at least 20% battery level is reached.

   R6 Apply the B fee.

   R7 Provide the money saving option.

7. **[G7] Keep cars always well distributed on the territory.**

   R1 Apply the B fee.

   R2 Periodically, if there is a non-uniform cars' distribution, send a notification to operators who will intervene.

   R3 View special parking areas.

   R4 Enable money saving option.

8. **[G8]Make sure that most of the cars are ready to use and in a good condition.**

   R1 When a technical issue is detected, a notification is sent to operators who will intervene and the car is switched to unavailable.

   R2 Make a report whenever a minor issue is reported by a user.

   R3 Periodically, an operator takes care of managing reported minor issues.

9. **[G9] Identify who the driver is for security purposes.**

R1 The user who unlocks the car doors must enter his personal password once in the car in order to be recognized.

R2 Every time the user wants to ignite the engine, he must be recognized.

R3 Every time the driver exits the car, the engine goes off after 15 seconds.

R4 In case of necessity, an operator can retrieve the identity of drivers of all rides on a specific car and report related information to recognized external authorities (like police and/or medical services).

R5 The user can ask for a new password.

- [R6] A ride can last for at most eight hours. After that, the public safety service is informed about a potential robbery.

2.2 Non-functional requirements

2.2.1 User experience

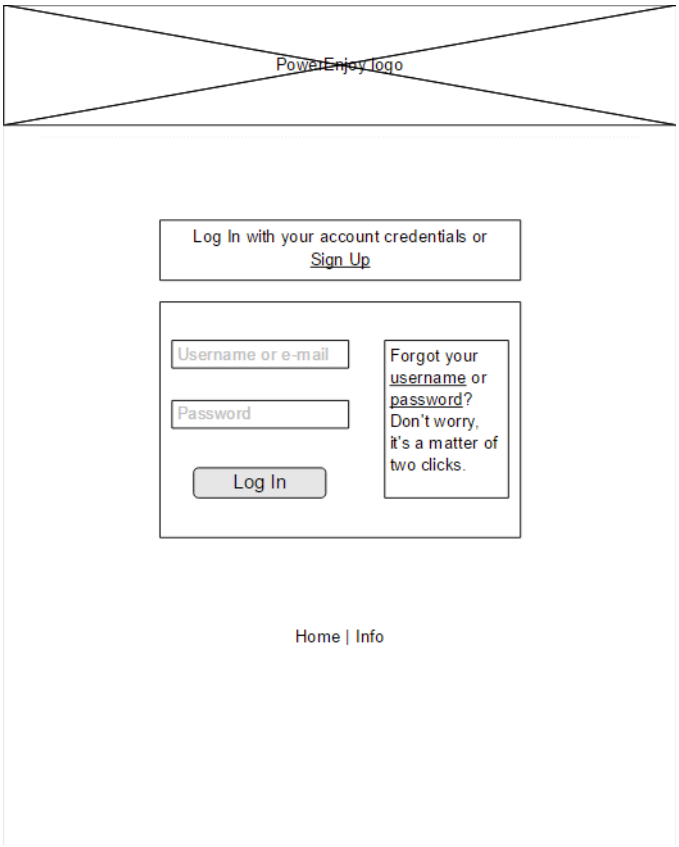
If the aim of PowerEnJoy is to be used daily, a great effort must be spent on the user experience, in particular we require the following characteristics:

- **Usability:** since the set of functionalities usable by the user is quite limited it should not be difficult to make them so intuitive that no documentation will be necessary to fully understand them at a first glance.
- **Security:** the manipulation of sensible data like payment methods requires a strong focus on security matters and the customers must be aware that they are placing their money in good hands.
- **Stability:** it's a major characteristic since the service must be available h24 7/7dd.
- **Look & feel:** an essential design is necessary to catch the customers' attention and bound them to the service.

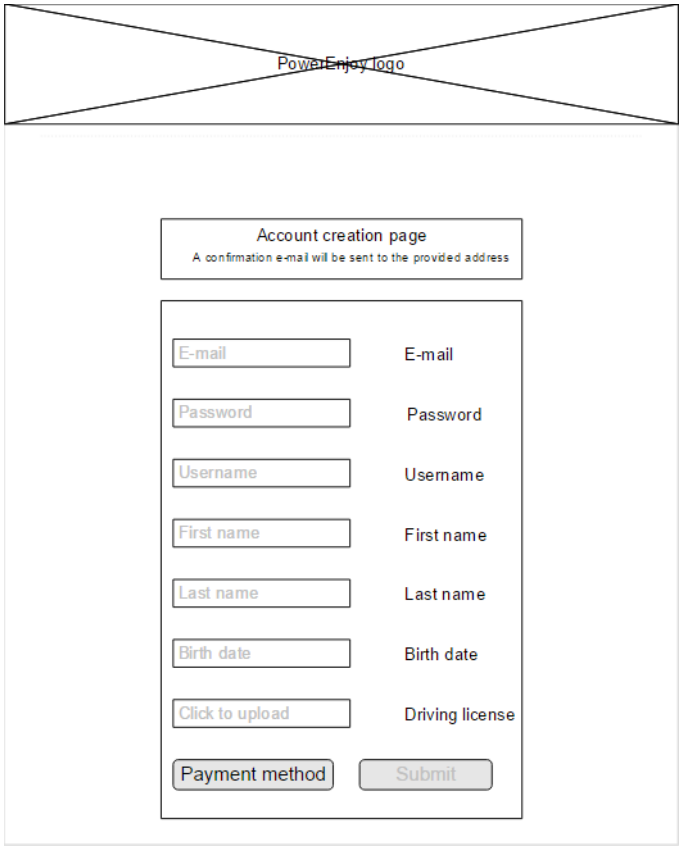
2.2.2 User Interface

In this paragraph, we go through the main features of the user interface. In particular, we concentrate on the app's most important pages and on the car screen, leaving the operator interface for a further development.

Login Page The login page should be intuitive, without any functionality but to sign in or create a new account. A "home" button is provided to get to the main page and surf the available cars (without login).

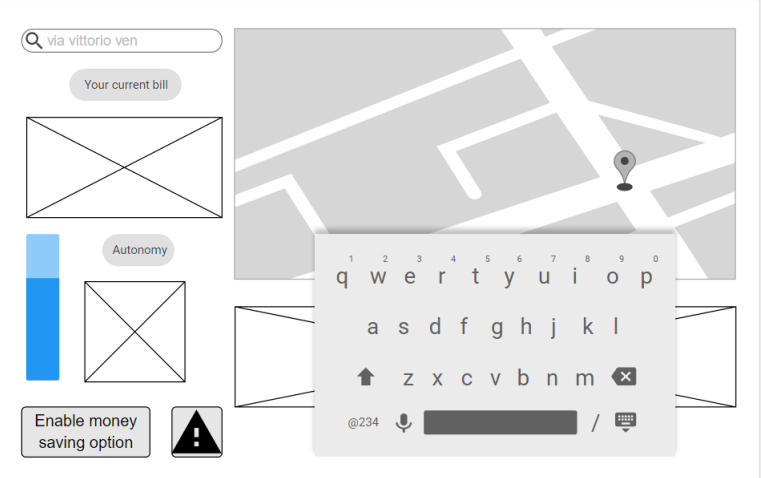

PowerEnjoy logo
Log In with your account credentials or
Sign Up
Username or e-mail
Password
Log In
Forgot your
username or
password?
Don't worry,
it's a matter of
two clicks.
Home | Info

Account creation page A scrollable page where new users can provide all the personal information necessary to create an account, comprehensive of a copy of the driving license. We must notice that in the following mockup not all the required information is represented (for example, a telephonic number is required). This was made only to make the representation clearer.



Main page The main page of the application. Through this page it is possible to search all the available cars in the selected zones and, once a car is selected, to make a reservation for that car and start the navigation tool (powered by Google). The car doors unlock button is also in the main page, so that a user who gets to the car using the navigation tool can unlock the doors from the same page. Unlogged users can get to the main page, but all the buttons below the map will be unavailable.

Car screen The following picture represents the car screen after it has been unlocked through the personal password. The navigation system is powered by Google Maps, so we will not go through its complete functionality. The car screen provides the driver with real-time information regarding the car's battery level and current money charge (excepted the application of fees and discounts).



2.2.3 Documentation

Here we report the list of the complete documentation that will be available ate the end of the deployment of the proposed system:

- **RASD**, Requirement Analysis and Specification Document, the present document, containing the description of goals and requirements along with a high level description of the proposed system. Technical properties are described through use case diagrams, UML diagrams and scenarios. The whole document.
- **DD**: Design Document, containing the functional description of the system.
- **Installation manual**: containing the instructions necessary for the deployment of the system.
- **User manual**: a complete description of the system from the point of view of the users.
- **Testing report**: the description of the tests used to confirm the system's reliability.
- **Project report**: the result of analysis conducted during the development of the system.

2.2.4 Considerations upon reliability and availability of the proposed system

The objective is to develop a system that ensures its service available H24 7/7. Since this goal is not obtainable in the real world, we plan to conduct all the necessary maintenance services during the night, when PowerEnJoy should be less requested, and with a large forewarning for customers. We plan to keep the maintenance services limited at 3 hours every two weeks, but these numbers will be revised in the precise analysis brought on in the Design Document. Reliability is a fundamental characteristic of the system to be developed. The application in write-intensive. Every reservation is critical, since it requires access and blocking of resources that are normally accessible by everyone. A recise reliability rate (reliability rate = mean time between failures) will be evaluated in the Design Document.

3 Scenario identifying

3.1 Scenario 1

Juan has planned a day trip to a museum with three friends of him. Since the museum is in the same city as them, Juan wants to get to the museum using PowerEnJoy, so he signs in (he already signed up three months before) and searches for a car. To be well prepared for the event, he also searches for the nearest special parking area through the app functionality. Half an hour before the meeting, Juan reserves a car and uses the "navigate to the car" functionality to get to the right location along with his friends. When everybody is there, he uses the "unlock doors" button to open the car doors, then they board together. Juan is asked to insert the password that was provided by the system, then he ignites the engine and searches for the destination through the car's screen. Juan presses the "money saving option" button on the screen to be informed of the upcoming discounts, then he starts driving to the selected location. When Juan arrives at the selected location he parks into a safe parking area, all the passengers get off the car and the doors lock. Ten minutes later, Juan is notified of the new invoice and he consults the final expenses for the ride.

3.2 Scenario 2

After Christmas holidays there are several cars that are located in areas that are very far from safe parking areas. A notification is sent to one of the operators of PowerEnJoy requesting his intervention to bring one of the vehicles back to the nearest special parking area. After the operator has completed the task another notification arrives claiming that a car was left unplugged with 5% battery, so he heads to the location of the vehicle and recharges it with the emergency kit he's provided with, then he boards on the car and brings it back to nearest safe parking area and makes sure the battery is at least 20% recharged.

3.3 Scenario 3

Julia has a reservation for a car but when she arrives at the vehicle's location she finds out the cabin is dirty. She presses the "report an issue" button on the car screen and cancels the reservation through the app. After a short time, an operator pushes the "show reported issues" button and takes the issue in charge. After he has opened a new paperwork, he uses the "find rides by plate number" to check who used the car the last time, then he heads at the car's location and resolves the issue requesting aid from the central. Being the problem not that grave, no notification is sent to the responsible user, but the operator still produces a report for the issue using the "create a report" functionality.

3.4 Scenario 4

One day a vandal breaks the glasses of a parked PowerEnJoy car. The system detects the issue and a notification is sent to an operator, who takes the problem in charge and immediately heads to the car's location. After he has checked the car's conditions he brings it to an affiliate mechanic to repair the damage.

3.5 Scenario 5

Anthon has boarded a car with his girlfriend to go to the cinema. They rush out the car to arrive in time for the show, but doing so Anthon parks in a safe parking area leaving the engine ignited. To make things worse, the car's battery is at 7% when Anthon and his girlfriend get off and the nearest power grid is 5km away from the car's location. In the interval, Anthon checks the invoice that was generated for the ride and he discovers not only that he has no right to any discount, but that he also must pay fee C.

3.6 Scenario 6

At three o'clock p.m. a notification for a technical issue (a pneumatic brake) appears on the page of an operator. He starts the procedure by marking the car as unavailable but, doing so, he finds out that the car was reserved in the meanwhile. The operator waits for the ride to end, then the car is finally marked as unavailable and he heads to its location. When the operator arrives, he discovers that the car's insides are not as clean as they should, so he resolves both the issues and writes down in the paperwork the actions he performed.

3.7 Scenario 7

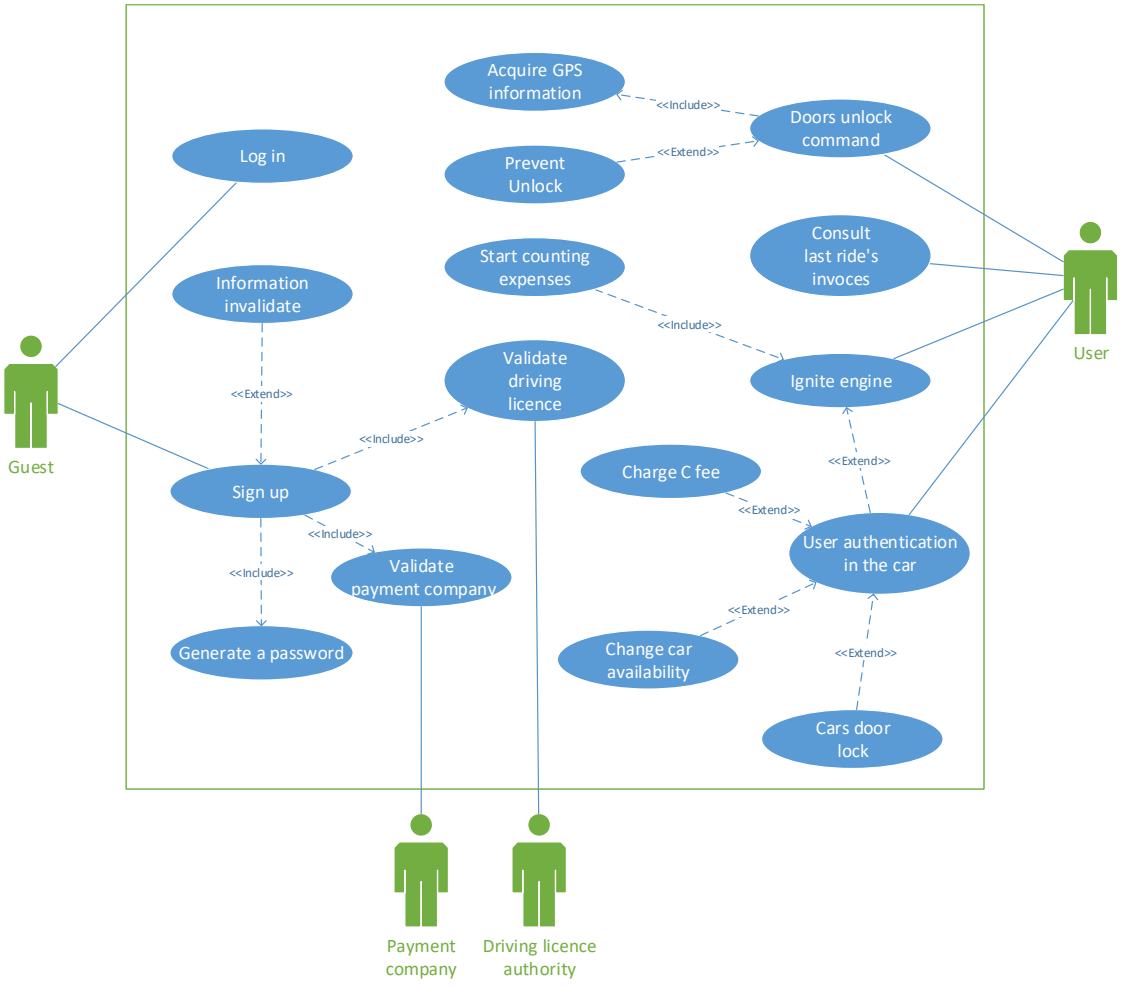
Maria boards a car and writes her personal password into the car screen in order to ignite the engine. While driving, Maria does not care about the road signal and she passes a crossroad while the traffic lights are red. The infraction is recorded through a photo of the car, and the consequent fine is sent to PowerEnJoy. An operator opens the practice and he searches for the ride in wich the infraction was committed. When he finds the right correspondence, he searches for Maria'informa in PowerEnJoy database and he sends the penalty to the address she indicated during the sign up phase.

4 UML models

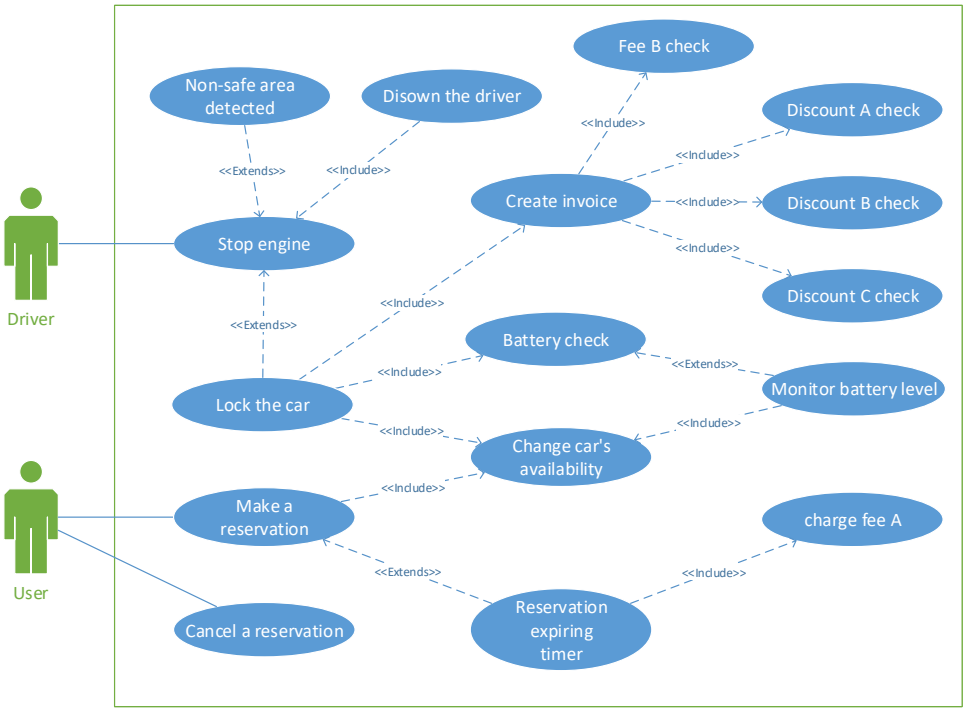
4.1 Use case diagrams

In the following section we present the actualization of the presented goals through the use of use case diagrams. We do not consider goal 5 in this part, since its actualization in made through the introduction of the discount A, which is hardly representable with this kind of diagram .

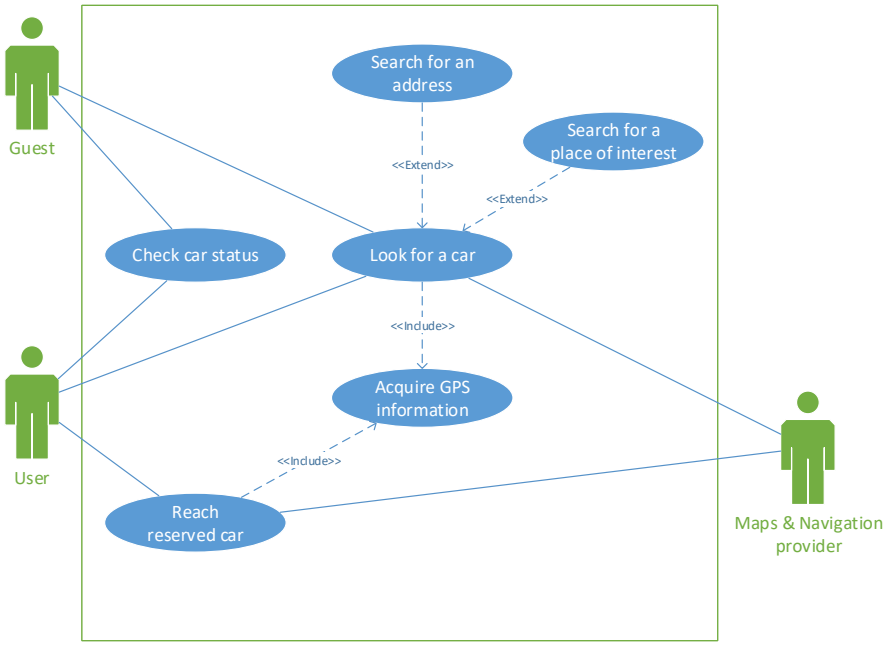
4.1.1 Use case diagram: goal 1



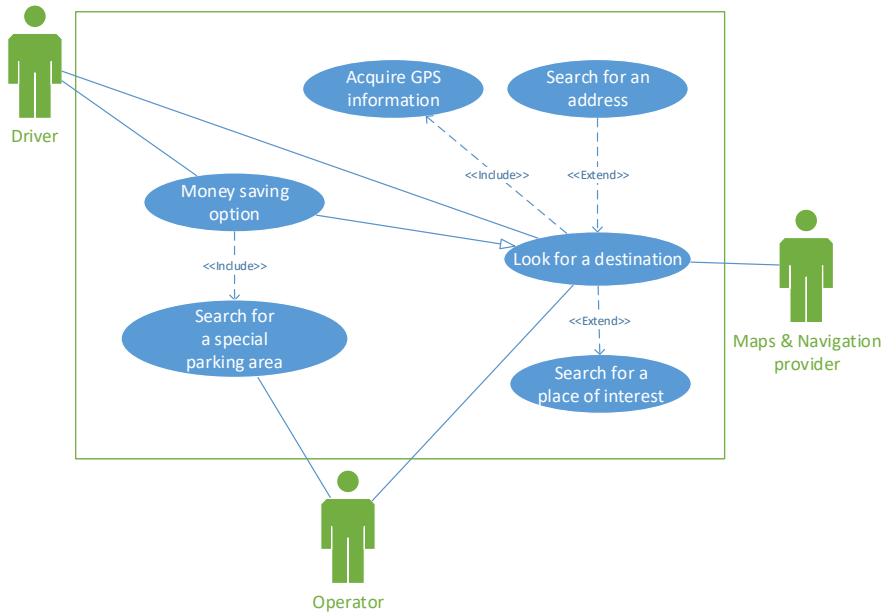
4.1.2 Use case diagram: goal 2



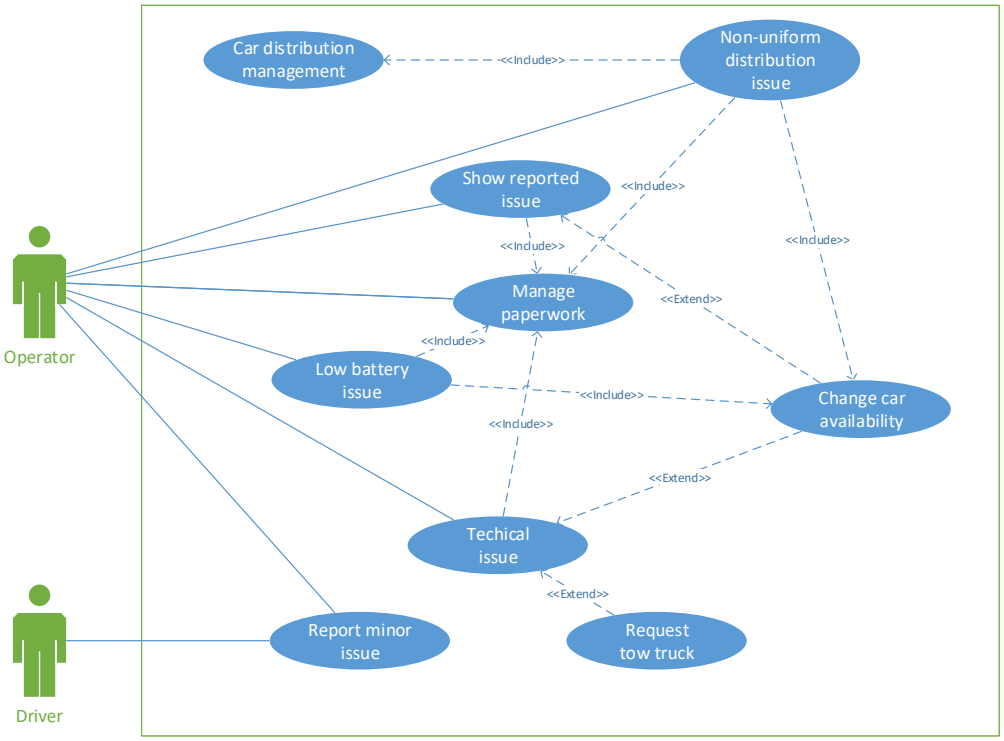
4.1.3 Use case diagram: goal 3



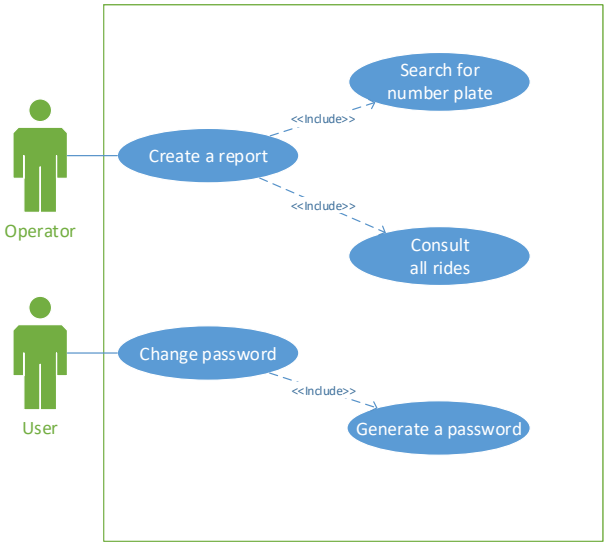
4.1.4 Use case diagram: goal 4



4.1.5 Use case diagram: goal 6, 7, 8



4.1.6 Use case diagram: goal 9

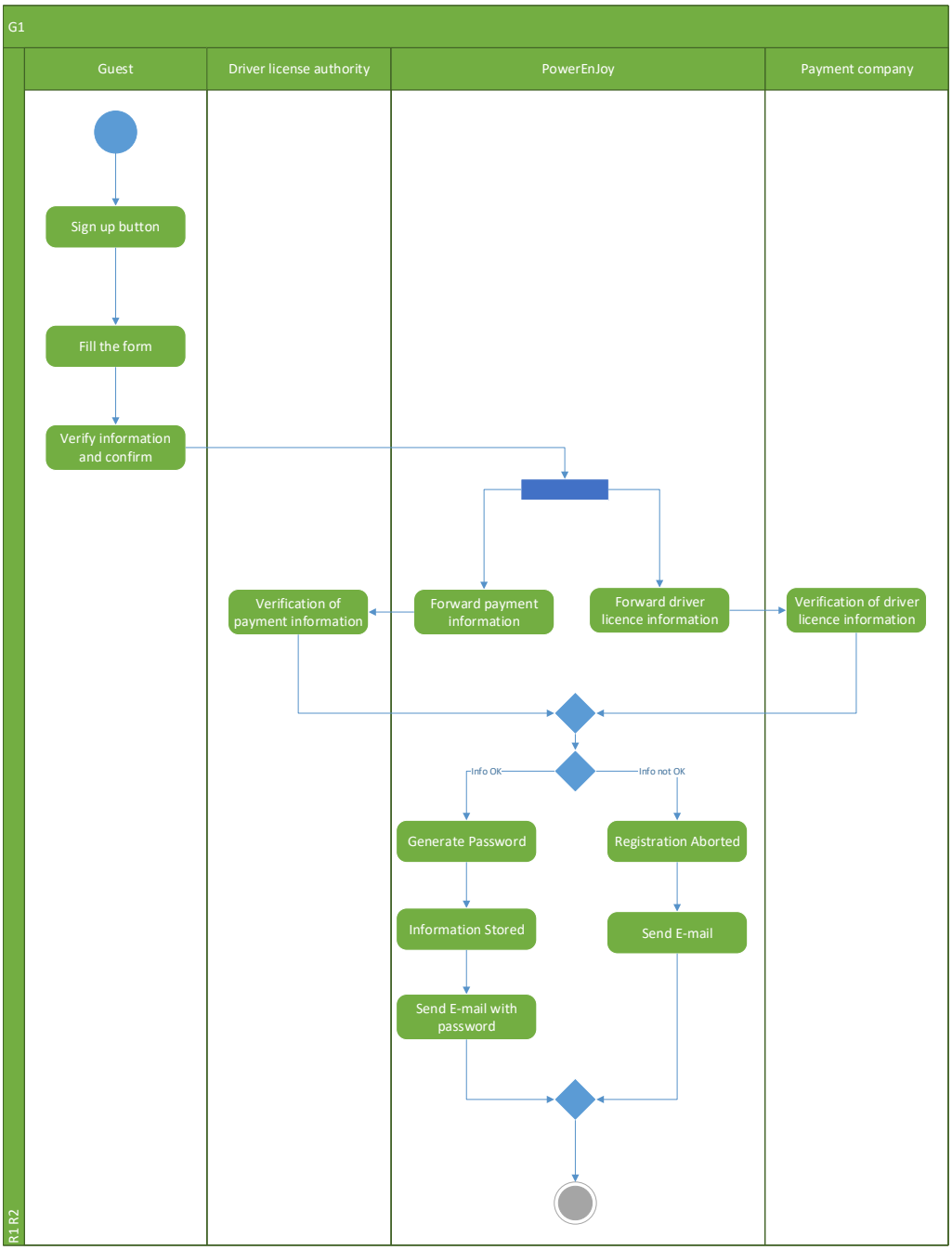


4.2 Use case description

In the following section we present a complete list of the use cases that may happen while using PowerEnJoy system, both from the perspective of the user and of the operator. The most significant use cases are correlated by sequence and activity diagrams that describe the events flow of the related use case.

4.2.1 Sign up

Actor	Guest
Goal	[G1] [R1] [R2]
Precondition	-
Event Flow	1. A guest on the home page presses the "sign up" button to start the registration process. 2. The guest fills the form with personal, payment and driving license information. 3. The guest verifies and confirms information entered. 4. PowerEnJoy forwards driving license information to the driver licensing authority which will validate them. 5. PowerEnJoy forwards payment information to the payment company which will validate them.
Post-Condition	The guest receives a password generated by PowerEnJoy to log in via e-mail. The guest can now log in PowerEnJoy. All the guest's information is stored.
Exception	a. A guest fills the form with information regarding another user. b. The driving license information has not been validated. c. The payment information has not been validated.
Exception Handling	(a), (b), (c) The signing up process is aborted and the guest is notified via e-mail.

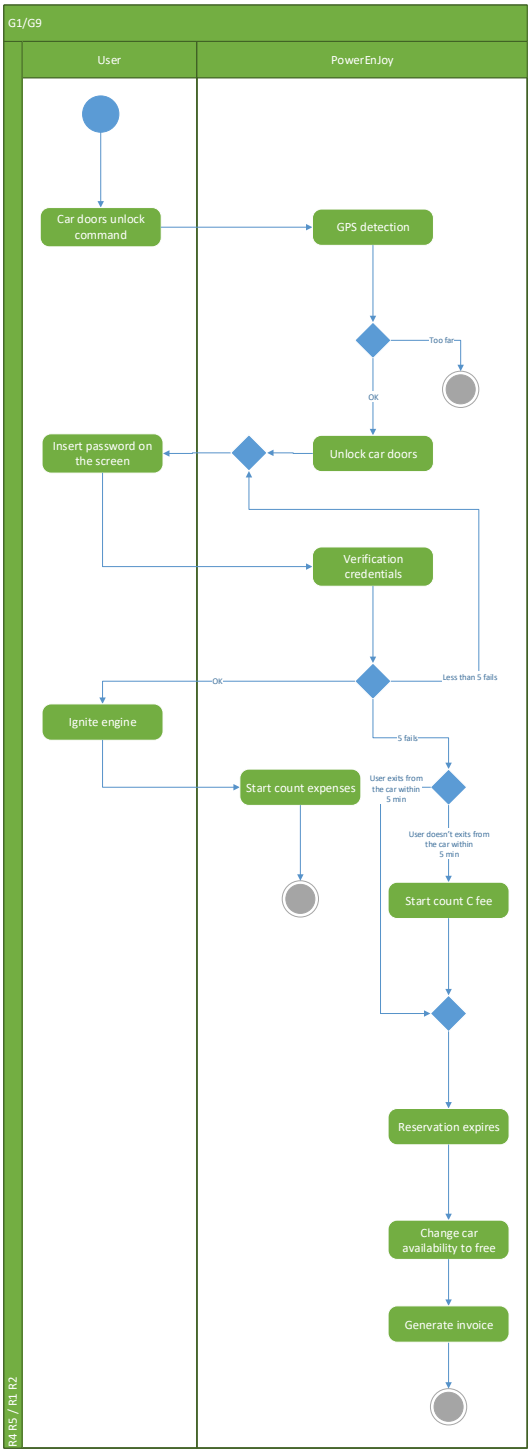

G1
Guest
Driver license authority
PowerEnJoy
Payment company
Sign up button
Fill the form
Verify information and confirm
Verification of payment information
Forward payment information
Forward driver licence information
Verification of driver licence information
Info OK
Info not OK
Generate Password
Registration Aborted
Information Stored
Send E-mail
Send E-mail with password
R1 R2

4.2.2 Log in

Actor	Guest/Operator
Goal	[G1][R3]
Precondition	-
Event Flow	1. A guest (operator) on the home page presses the "log in" button to use PowerEnJoy's features. 2. The guest (operator) fills basic personal information and the password given by PowerEnJoy
Post-Condition	The guest is logged in PowerEnJoy. The guest has become a user (or the operator is recognized).
Exception	a. Either basic personal information or password is wrong.
Exception Handling	(a) Ask the user to refill both basic personal information and the password.

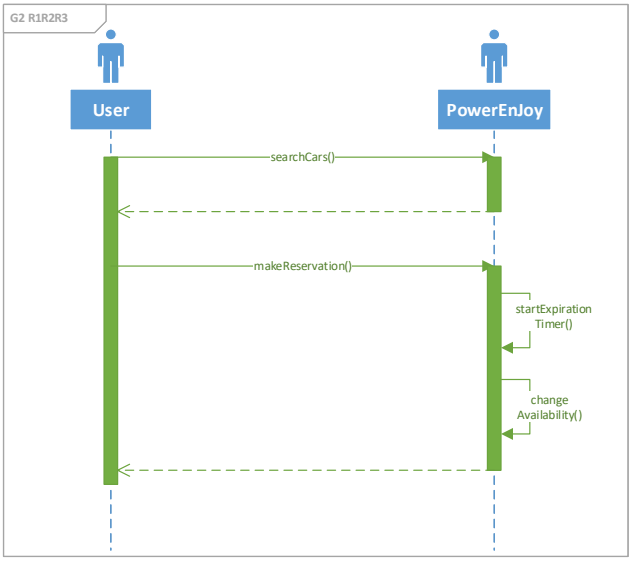
4.2.3 Start ride

Actor	User
Goal	[G1] [R4] [R5] [G9] [R1] [R2]
Precondition	The user has an active reservation for a car.
Event Flow	1. The user launches the car doors unlock command. 2. PowerEnJoy detects the user's position using the GPS. 3. PowerEnJoy unlocks car doors. 4. The user enters his password through the car's screen. 5. PowerEnJoy recognizes the user. 6. The user ignites the engine for the first time.
Post-Condition	The ride starts. The user becomes a driver. Start counting expenses for a given amount of money per minute. The driver can drive the car.
Exception	a. The user is too far from the car. b. The wrong password is inserted for five times. c. The car is linked to a power turret.
Exception Handling	(a) Prevent car doors unlock. (b) PowerEnJoy prevents further password insertion and the user has the possibility to exit the car within five minutes from the car doors unlock otherwise he starts paying the C fee. Then, the car doors lock, the reservation expires, the car status is switched to available and an invoice reporting expenses is generated. (c) Once the car doors are unlocked, the user has also to unplug the car from the power turret, otherwise the engine does not ignite.


G1/G9
User
PowerEnJoy
Car doors unlock command
GPS detection
Too far
OK
Insert password on the screen
Unlock car doors
Verification credentials
Less than 5 fails
OK
Ignite engine
5 fails
Start count expenses
User exits from the car within 5 min
User doesn't exits from the car within 5 min
Start count C fee
Reservation expires
Change car availability to free
Generate invoice
R4 R5 / R1 R2

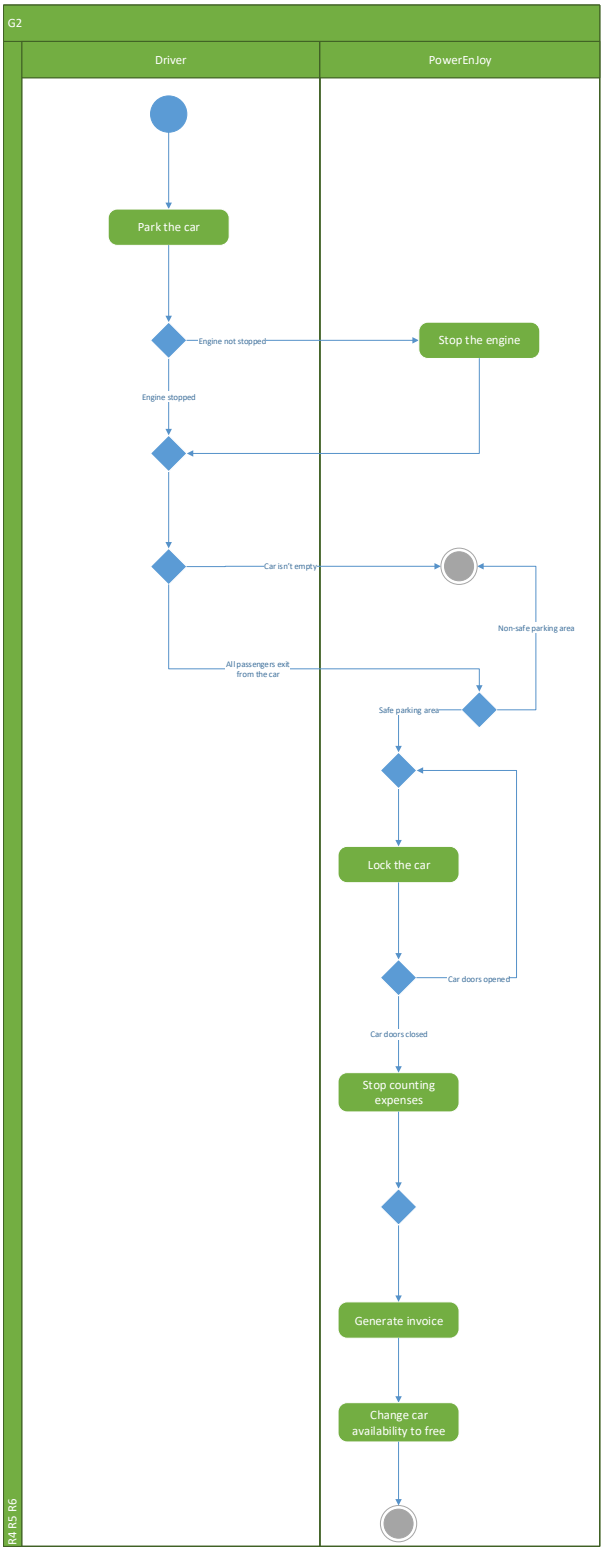
4.2.4 Make reservation

Actor	User
Goal	[G2] [R1] [R2] [R3]
Precondition	The user has found an available car.
Event Flow	1. The user presses the "make a reservation" button in the app. 2. Reservation expiring timer starts. 3. The car is marked as unavailable.
Post-Condition	The user has reserved a car.
Exception	a. Another user makes a reservation for the same car in the very short moment before so the car is actually unavailable. b. The user does not start the ride within one hour. c. The user cancels the reservation.
Exception Handling	(a) The user may find another available car. (b) The user has to pay the A fee and the reservation for the car expires. (c) The reservation is canceled.



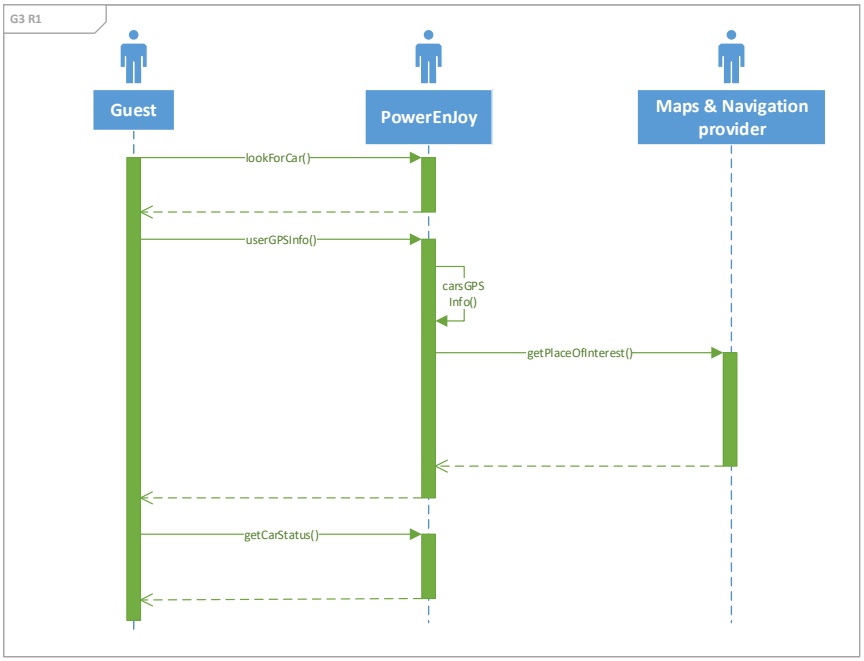
4.2.5 End ride

Actor	Driver
Goal	[G2] [R4] [R5] [R6]
Precondition	-
Event Flow	1. The driver parks into a safe parking area. 2. The driver turns off the engine definitively. 3. Every passenger gets off the car. 4. All doors are closed.
Post-Condition	PowerEnJoy locks car doors within one minute. The ride ends. The driver returns to be considered a user. PowerEnJoy stops charging expenses. PowerEnJoy generates the invoice corresponding to the ride. The car is marked as available.
Exception	a. The driver parks in a non-safe parking area. b. One or more passengers do not get off the car. c. Not all doors are closed. d. The driver does not turn off the engine. e. The car has almost empty battery. f. The car is parked in a special parking area.
Exception Handling	(a), (b), (c) The car doors does not unlock and the ride does not terminate. (d) The engine will be automatically switched off within 20 seconds as the driver exit. (e) Car doors get locked, the ride ends but the car is marked as unavailable. PowerEnJoy charges the B fee unless it is parked in a special parking area. (f) As the event flow below. In addition, the driver has to plug the car to the power turret otherwise the ride will not end. Then, all post-conditions above occur.


G2
Driver
PowerEnJoy
Park the car
Engine not stopped
Stop the engine
Engine stopped
Car isn't empty
Non-safe parking area
All passengers exit from the car
Safe parking area
Lock the car
Car doors opened
Car doors closed
Stop counting expenses
Generate invoice
Change car availability to free
R4 R5 R6

4.2.6 Look for a car

Actor	Guest / User
Goal	[G3] [R1]
Precondition	-
Event Flow	1. A guest or a user on the home page presses the "look for a car" button to find a car. 2. The guest or the user fills the search filter searching for places of interest, address or actual position. 3. The guest or the user may consult cars' status.
Post-Condition	PowerEnJoy finds out all the cars which match the search filter. Cars' status is available to be consulted.
Exception	-
Exception Handling	-



4.2.7 Navigate to a car

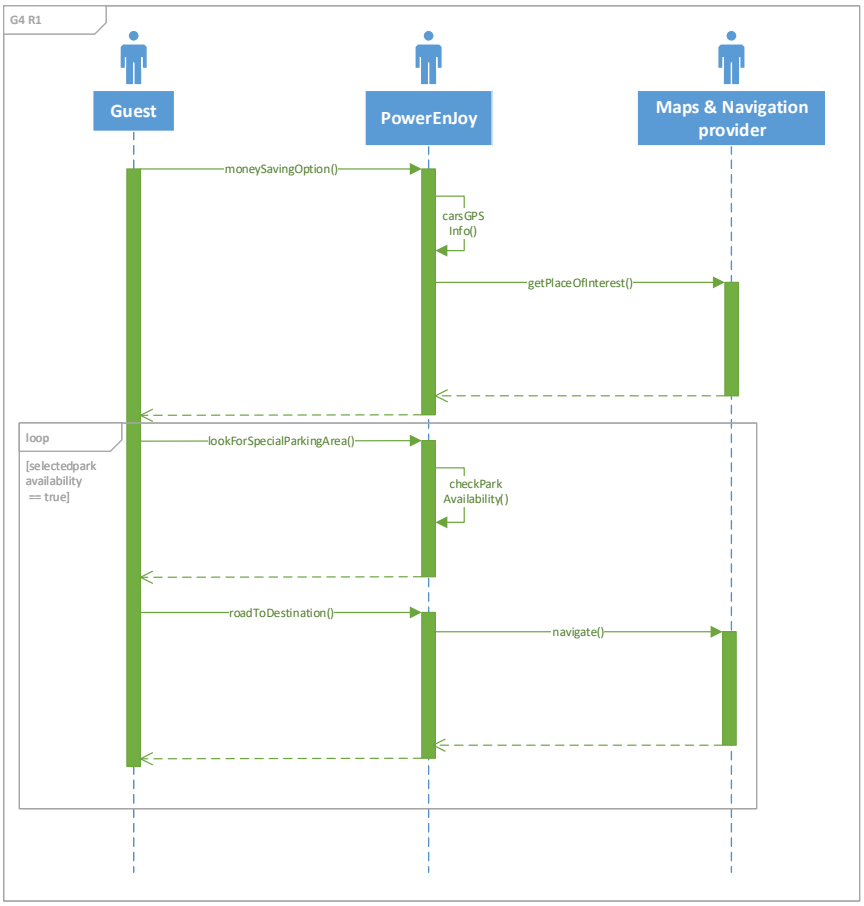
Actor	User
Goal	[G3] [R2]
Precondition	The user has an active reservation for a car.
Event Flow	1. The user presses the "navigate to the car" button to reach the car. 2. The user selects a path according to his necessity.
Post-Condition	PowerEnJoy provides navigation tips to the user in order to reach the reserved car.
Exception	-
Exception Handling	-

4.2.8 Navigate to a location

Actor	Driver / Operator
Goal	[G4] [R1]
Precondition	-
Event Flow	1. A driver/operator presses the "look for a destination" text area in order to navigate to a specific location. 2. The driver/operator fills the search filter according to places of interest or a specific address. 3. The driver selects a path according to his necessity.
Post-Condition	PowerEnJoy provides navigation tips to reach the target location.
Exception	-
Exception Handling	-

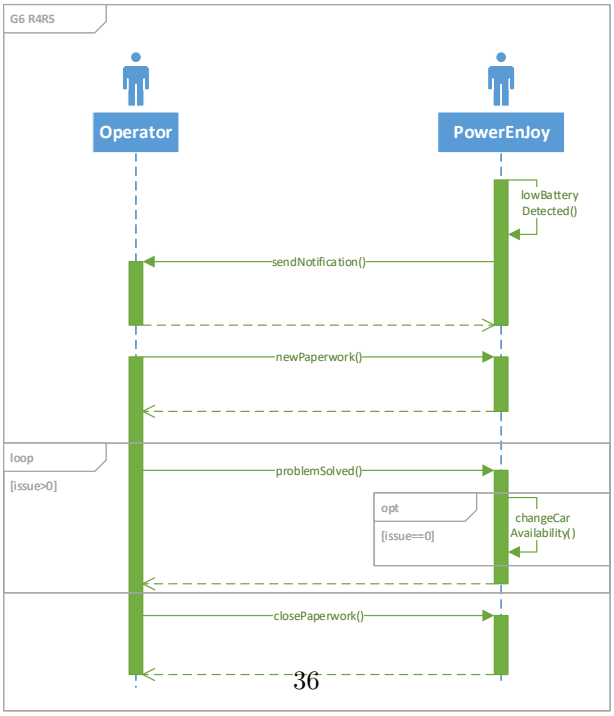
4.2.9 Use the money saving option

Actor	Driver
Goal	[G4] [R2]
Precondition	-
Event Flow	1. A driver on the screen presses the "money saving option" button. 2. The driver fills the search filter according to places of interest or a specific address. 3. PowerEnJoy selects a special parking area amongst the ones near to the target location in order to avoid the overcrowding of some parking areas.
Post-Condition	PowerEnJoy provides navigation tips to the driver in order to reach the nearest special parking area from the target location.
Exception	a. While reaching the special parking area, it may happen that the target special parking area becomes full.
Exception Handling	(a) PowerEnJoy notifies the driver that the selected special parking area is not available anymore. Then, it evaluates another special parking area target.


G4 R1
Guest
PowerEnJoy
Maps & Navigation provider
moneySavingOption()
carsGPS Info()
getPlaceOfInterest()
loop
[selectedpark availability == true]
lookForSpecialParkingArea()
checkPark Availability()
roadToDestination()
navigate()

4.2.10 Solve low battery issue

Actor	Operator
Goal	[G6] [R4] [R5]
Precondition	PowerEnJoy detects an almost empty battery level car among those which are not charging and it notifies the operator.
Event Flow	1. The operator accepts the notification. 2. The operator opens a new paperwork. 3. PowerEnJoy shows cars with low battery level. 4. The operator picks the car through a by-pass. 5. The operator moves the car to a special parking area. 6. The operator plugs the car to a power grid station. 7. PowerEnJoy will restore car's availability when 20% level of battery is reached. 8. The operator closes the paperwork.
Post-Condition	A car is placed in a special parking area and it is charging.
Exception	a. The car has a very low battery level and the operator can not move it to a special parking area. b. In addition, the car has either a technical issue or a minor issue.
Exception Handling	(a) The operator reaches the car bringing with him a supply battery and plugs it. After an enough battery charge, the operator takes care of moving the car to a special parking area, to plug it and to restore the car's availability. (b) The operator will also take care of other problems according to the related routine procedure.

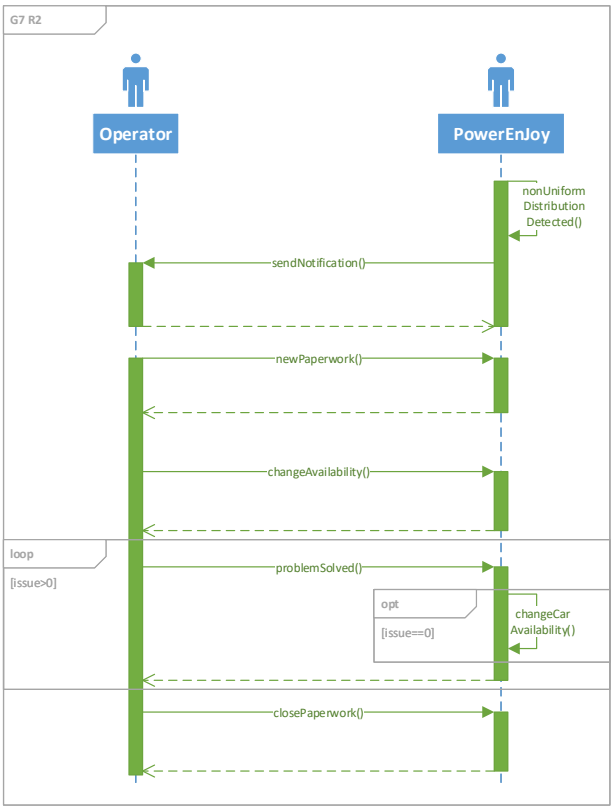


4.2.11 Show special parking areas

Actor	Guest / User / Driver / Operator
Goal	[G6] [R1]
Precondition	-
Event Flow	1. An actor on the home page clicks on the “show special parking area” button.
Post-Condition	PowerEnJoy provides a list of all special parking areas.
Exception	-
Exception Handling	-

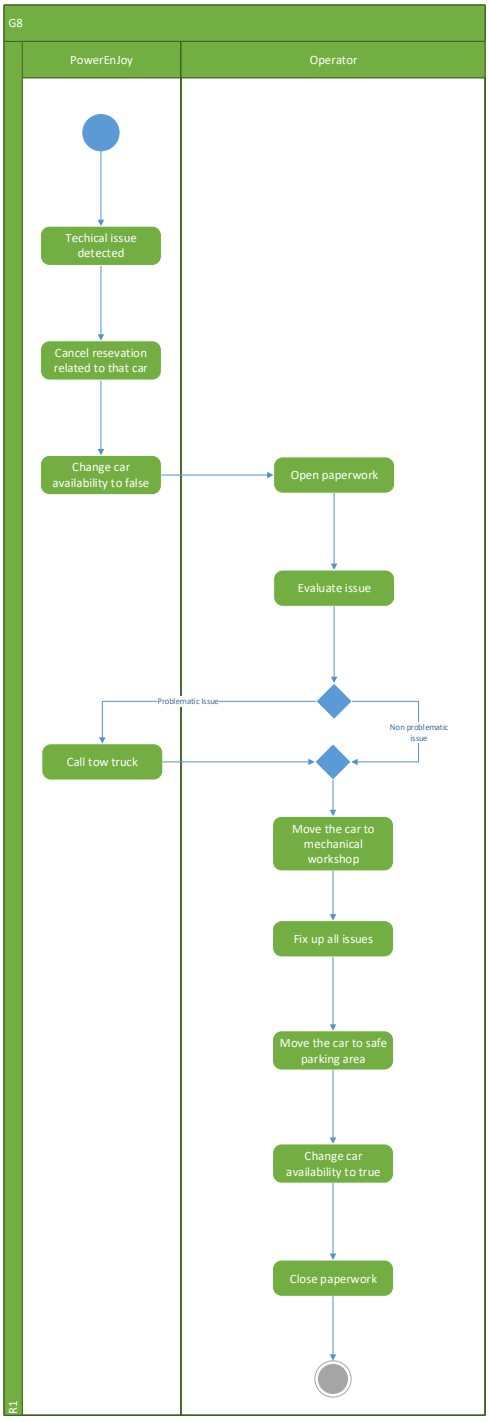
4.2.12 Solve non-uniform cars distribution issue

Actor	Operator
Goal	[G7] [R2]
Precondition	PowerEnJoy detects a non-uniform cars' distribution among those which are not reserved and it notifies the operator.
Event Flow	1. The operator receives notification. 2. The operator opens a new paperwork to manage the issue. 3. PowerEnJoy shows cars to be potentially re-distributed. 4. The operator selects a car and manually switches it to unavailable. 5. PowerEnJoy suggests a safe parking area where the car can be left. 6. The operator picks the car through a by-pass. 7. The operator moves the car to the suggested safe parking area. 8. The operator manually switches the car as available. 9. The operator closes the paperwork.
Post-Condition	The car is placed in a non crowded safe parking area.
Exception	a. A user makes a reservation for a car which has been shown in the list of cars to be potentially re-distributed before the operator selects it and marks it as unavailable. b. The car has either a low battery level, a technical issue or a minor issue.
Exception Handling	(a) PowerEnJoy warns the operator that the selected car has been reserved by a user. So, the operator is prompted to the list of cars to be potentially re-distributed. (b) The operator will also take care of other problems according to the related routine procedure.


G7 R2
Operator
PowerEnJoy
nonUniform Distribution Detected()
sendNotification()
newPaperwork()
changeAvailability()
loop
[issue>0]
problemSolved()
opt
[issue==0]
changeCar Availability()
closePaperwork()

4.2.13 Solve technical issue

Actor	Operator
Goal	[G8] [R1]
Precondition	PowerEnJoy detects a technical issue and it sends a notification to an operator.
Event Flow	1. PowerEnJoy marks the car as unavailable. 2. The operator accepts the notification. 3. The operator opens a new paperwork to manage the issue. 4. The operator evaluates the issue relevance. 5. The operator picks the car through a by-pass. 6. The operator, once in the car, may either launch the external tow truck service or move the car to the affiliate mechanical workshop according to the issue relevance. 7. After having fixed the issue, the operator brings the car to a safe parking area. 8. The operator restores car availability. 9. The operator closes the paperwork.
Post-Condition	The car is ready to serve.
Exception	a. the technical issue is detected while a user has a reservation but the ride is not startet yet. b. A user makes a reservation before PowerEnJoy has marked the car as unavailable. c. The technical issue is detected during a ride. d. In addition, the car has either a low battery level or a minor issue.
Exception Handling	(a), (b) PowerEnJoy cancels the reservation and notifies the user through the app. (c) PowerEnJoy exhorts the driver to end the ride. (d) The operator will also take care of other problems according to the related routine procedure.

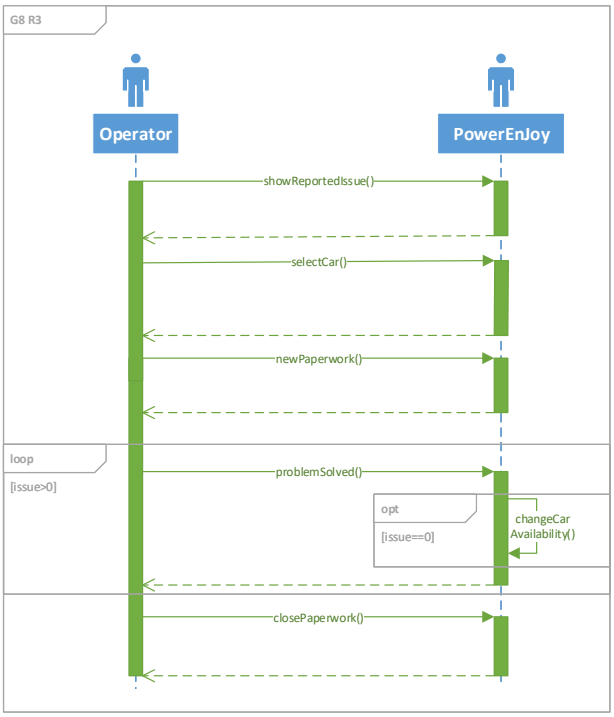

G8
PowerEnJoy
Operator
Techical issue detected
Cancel resevation related to that car
Change car availability to false
Open paperwork
Evaluate issue
Problematic issue
Non problematic issue
Call tow truck
Move the car to mechanical workshop
Fix up all issues
Move the car to safe parking area
Change car availability to true
Close paperwork
R1

4.2.14 Report minor issue

Actor	Driver
Goal	[G8] [R2]
Precondition	-
Event Flow	1. The driver pushes the "report an issue" button through the car screen. 2. The driver fills relevant fields regarding the observed issue.
Post-Condition	The report is saved.
Exception	-
Exception Handling	-

4.2.15 Solve minor issue

Actor	Operator
Goal	[G8] [R3]
Precondition	The car is not reserved.
Event Flow	1. The operator pushes the "show reported issues" button. 2. The operator selects all minor issues regarding a single car. 3. The operator opens a new paperwork to manage the car minor issues. 4. The operator switches the car as unavailable. 5. The operator picks the car through a by-pass. 6. The operator takes care of solving the minor issue. 7. The operator brings the car to a safe parking area. 8. The operator restores car availability. 9. The operator closes the paperwork.
Post-Condition	The car is clean and in a good condition.
Exception	a. A user makes a reservation for a car which has been shown in the list of cars with minor issues before the operator marks that car as unavailable. b. In addition, the car has either a low battery level or a technical issue.
Exception Handling	(a) PowerEnJoy warns the operator that the selected car has been reserved by a user. So, the operator is prompted to the list of cars with minor issues. (b) The operator will also take care of other problems according to the related routine procedure.


G8 R3
Operator
PowerEnJoy
showReportedIssue()
selectCar()
newPaperwork()
loop
[issue>0]
problemSolved()
opt
[issue==0]
changeCar Availability()
closePaperwork()

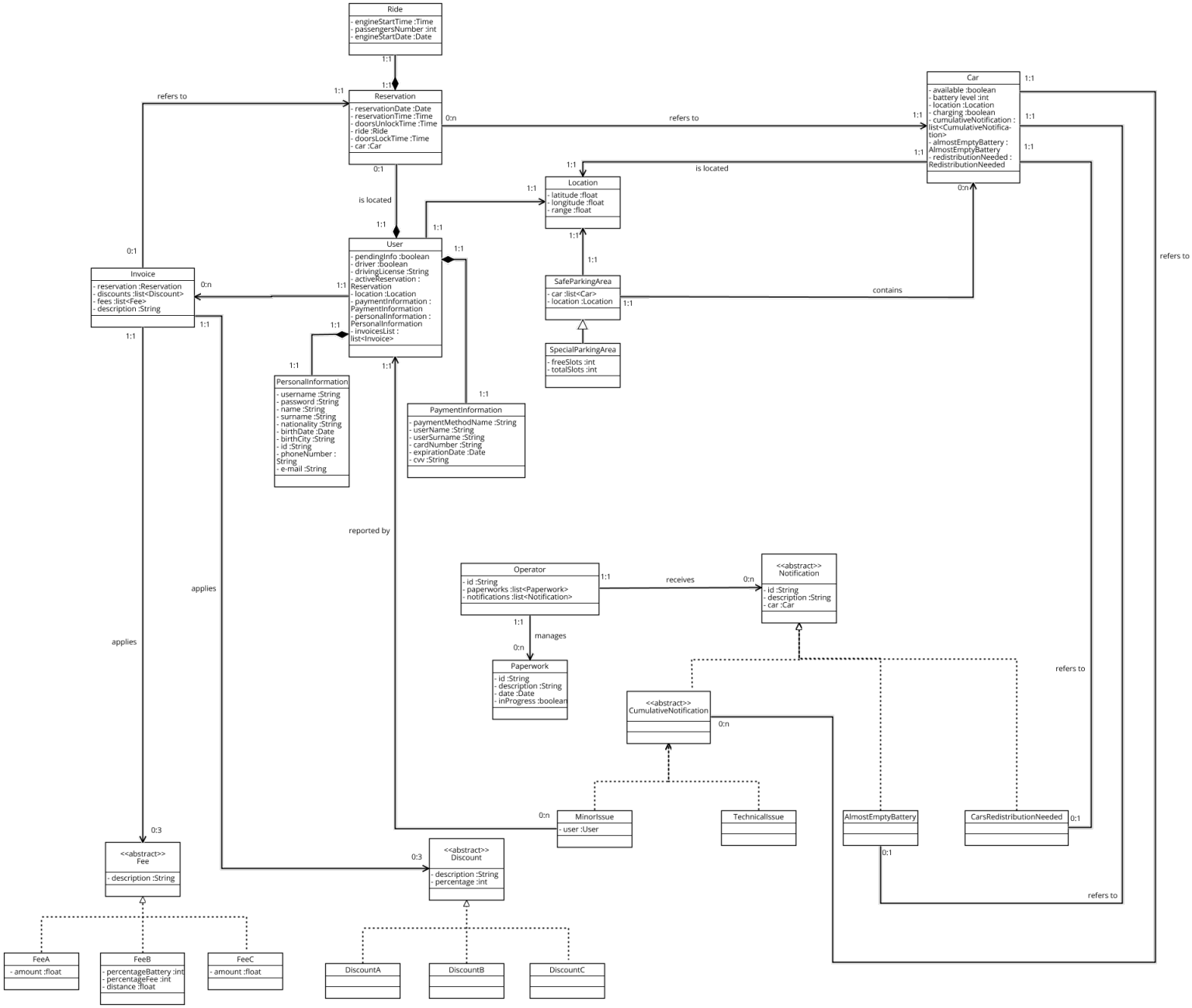
4.2.16 Consult a ride's information

Actor	Operator
Goal	[G9] [R4]
Precondition	-
Event Flow	1. The operator pushes the "show rides" button. 2. The operator selects details about a ride. 3. The operator selects the "create legal report" button in order to create a summary document about that ride.
Post-Condition	A report of the ride is created and sent via e-mail to recognized external authorities (like police and/or medical services).
Exception	-
Exception Handling	-

4.2.17 Generate a new password

Actor	User
Goal	[G9] [R5]
Precondition	-
Event Flow	1. The user pushes the "generate a new password" button on his profile settings.
Post-Condition	PowerEnJoy sends a new personal password associated to the user via e-mail.
Exception	-
Exception Handling	-

4.3 Class diagram



5 Alloy modeling

5.1 Signatures

```
sig Area {
}

sig Position {
}

abstract sig Notification {
    car: Car
}

abstract sig CumulativeNotification extends Notification {
}

sig TechnicalIssue extends CumulativeNotification {
}

sig MinorIssue extends CumulativeNotification {
    user: User
}

sig AlmostEmptyBatteryIssue extends Notification {
}

sig CarRedistributionNeeded extends Notification {
}

sig Car {
    position: Position,
    area: Area,
    cumulativeNotifications: set CumulativeNotification,
    almostEmptyBattery: lone AlmostEmptyBatteryIssue,
    redistributionNeeded: lone CarRedistributionNeeded
}

sig User {
    position: Position,
    personalInformation: PersonalInformation,
    paymentInformation: PaymentInformation,
    activeReservation: lone Reservation,
    invoices: set Invoice
}

sig PersonalInformation {
}

sig PaymentInformation {
}

sig Reservation {
    car: Car,
    ride: lone Ride
}

sig Ride {
    passengersNumber: Int,
} {
    passengersNumber >= 1 and passengersNumber <= 5
}

sig SafeParkingArea {
    area: Area,
    cars: set Car
}

sig SpecialParkingArea extends SafeParkingArea {
}
```

```
sig AFee extends Fee {
}

sig BFee extends Fee {
}

sig CFee extends Fee {
}

abstract sig Fee {
}

sig ADiscount extends Discount {
}

sig BDiscount extends Discount {
}

sig CDiscount extends Discount {
}

abstract sig Discount {
}

sig Invoice {
    fees: set Fee,
    discounts: set Discount
}

sig Paperwork {
}

sig Operator {
    paperworks: set Paperwork,
    notifications: set Notification
}
```

5.2 Facts

```
fact {

    //each notification is related to the corresponding car and the car has the set of his notifications
    all c: Car, n: CumulativeNotification | (n in c.cumulativeNotifications and c = n.car)
        or (n not in c.cumulativeNotifications and c != n.car)

    //each almost empty  issue is related to the corresponding car and the car has his almost empty battery issue
    all c: Car, empty: AlmostEmptyBatteryIssue | (c.almostEmptyBattery = empty and empty.car = c)
        or (c.almostEmptyBattery != empty and empty.car != c)

    // each redistribution needed notification is related to exactly one car
    all c: Car, r: CarRedistributionNeeded | (c.redistributionNeeded = r and r.car = c)
        or (c.redistributionNeeded != r and r.car != c)

    //no car with same position
    no c1, c2: Car | c1 != c2 and c1.position = c2.position

    //no user cannot have personal information and no user with the same personal information
    all personalInfo: PersonalInformation | (one u: User | u.personalInformation = personalInfo)

    //no user cannot have payment information
    all paymentInfo: PaymentInformation | (one u: User | u.paymentInformation = paymentInfo)

    //no ride without reservation and no ride of the same reservation
    all r: Ride | (one res: Reservation | res.ride = r)

    //two reservations have different cars
    no r1, r2: Reservation, c: Car | r1 != r2 and r1.car = c and r2.car = c

    //no reservation for technical issued cars
    all t: TechnicalIssue | (all r: Reservation | t not in r.car.cumulativeNotifications)

    //no reservation for almost empty battery cars
    all empty: AlmostEmptyBatteryIssue | (all r: Reservation | empty != r.car.almostEmptyBattery)

    //no reservation without user and no user with the same active reservation
    all r: Reservation | (one u: User | r = u.activeReservation)

    //riding cars are at the same position of the driver
    all ri: Ride | (one r: Reservation | (one u: User | r = u.activeReservation and r.car.position = u.position and r.ride = ri))

    //riding cars have not cars redistribution needed issue
    all redistrib: CarRedistributionNeeded | (all r: Reservation | redistrib != r.car.redistributionNeeded)

    //no safe parking area in the same area
    no park1, park2: SafeParkingArea | park1 != park2 and park1.area = park2.area

    //two safe parking areas has different cars
    no park1, park2: SafeParkingArea, c: Car | park1 != park2 and c in park1.cars and c in park2.cars

    //a car in a safe parking area have the same area
    no c1, c2: Car, park: SafeParkingArea | (c1 in park.cars and c2 in park.cars)
        and not (c1.area = c2.area and c1.area = park.area)

    //a car which is not in a ride must be in a safe parking area
    all c: Car, r: Reservation | no (r.ride & Ride) and r.car = c => c in SafeParkingArea.cars
    all c: Car | no (c & Reservation.car) implies c in SafeParkingArea.cars

    //two users have two different invoices
    no u1, u2: User, i: Invoice | u1 != u2 and i in u1.invoices and i in u2.invoices

    //no invoice without user
    all i: Invoice | (one u: User | i in u.invoices)

    //no fee without invoice
    all f: Fee | (one i: Invoice | f in i.fees)

    //no discount without invoice
    all d: Discount | (one i: Invoice | d in i.discounts)
```

```
    //AFee only if no other fee/discount
    no a: AFee, i: Invoice | a in i.fees and ((i.fees - a) != none or i.discounts != none)

    //no C discount and B fee in the same invoice
    no c: CDiscount, b: BFee, i: Invoice | c in i.discounts and b in i.fees

    //no same type of fee in the same invoice
    all i: Invoice | #(AFee & i.fees) <= 1 and #(BFee & i.fees) <= 1 and #(CFee & i.fees) <= 1

    //no same type of discount in the same invoice
    all i: Invoice | #(ADiscount & i.discounts) <= 1 and #(BDiscount & i.discounts) <= 1
        and #(CDiscount & i.discounts) <= 1

    //minor issue only if reservation or invoice for that user
    all m: MinorIssue | m.user.activeReservation.car = m.car or m.user.invoices != none

    //two paperworks have different operator
    no p1, p2: Paperwork, o: Operator | p1 != p2 and p1 in o.paperworks and p2 in o.paperworks

    //no paperwork without operator
    all p: Paperwork | (one o: Operator | p in o.paperworks)

    //no notification without operator
    all n: Notification | (one o: Operator | n in o.notifications)

    //all notifications are sent to all operators
    all o: Operator | o.notifications = Notification

    //no positions without users or cars
    User.position + Car.position = Position

    //no areas without cars or parking areas
    Car.area + SafeParkingArea.area = Area

}
```

5.3 Assertions

```
assert makingReservation {
    all u, u': User, r: Reservation | u.activeReservation = none and makeReservation[u, u', r]
        implies u'.activeReservation = r
}
check makingReservation

assert endingReservation {
    all u, u': User, r: Reservation, i: Invoice | u.activeReservation = r and i not in u.invoices and endReservation[u, u', r, i]
        implies u'.activeReservation = none and u'.activeReservation.ride = none and i in u'.invoices
}
check endingReservation

assert startingRide {
    all u, u': User, r: Reservation, ri: Ride | u.activeReservation = r and u.activeReservation.ride = none and startRide[u, u', r, ri]
        implies u'.activeReservation.ride = ri
}
check startingRide

assert endingRide {
    all u, u': User, r: Reservation, i: Invoice, ri: Ride | u.activeReservation = r and u.activeReservation.ride = ri and endRide[u, u', r, ri, i]
        implies u'.activeReservation.ride = none and endReservation[u, u', r, i]
}
check endingRide

assert reportingMinorIssue {
    all u: User, c, c': Car, m: MinorIssue |
        u.activeReservation.car = c and m not in c.cumulativeNotifications and reportMinorIssue[c, c', m] implies
            m in c'.cumulativeNotifications and m in Operator.notifications
}
check reportingMinorIssue

assert applyingBFee {
    all i, i': Invoice, b: BFee, u, u': User, r: Reservation, c: Car, ri: Ride |
        u.activeReservation = r and u.activeReservation.ride = ri and i not in u.invoices and endRide[u, u', r, ri, i]
        and r.car = c and c.almostEmptyBattery in AlmostEmptyBatteryIssue and applyBFee[i, i', b]
            implies b in i'.fees
}
check applyingBFee

assert applyingADiscount {
    all i, i': Invoice, a: ADiscount, u, u':User, r: Reservation, ri: Ride | u.activeReservation = r
        and u.activeReservation.ride = ri and i not in u.invoices and endRide[u, u', r, ri, i]
        and r.ride.passengersNumber >= 3 and applyADiscount[i, i', a]
            implies a in i'.discounts
}
check applyingADiscount

assert applyingCDiscount {
    all i, i': Invoice, c: CDiscount, u, u': User, r: Reservation, ri: Ride, park: SpecialParkingArea |
        u.activeReservation = r and u.activeReservation.ride = ri and i not in u.invoices
        and endRide[u, u', r, ri, i] and r.car in park.cars and applyCDiscount[i, i', c]
            implies c in i'.discounts
}
check applyingCDiscount
```

5.4 Predicates

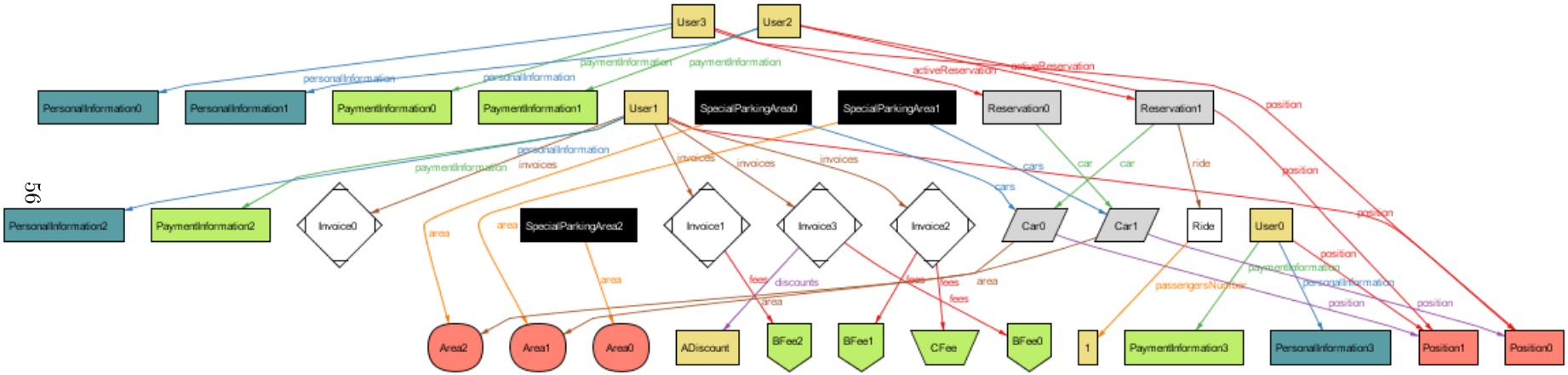
```
pred makeReservation(u, u': User, r: Reservation) {
    u'.activeReservation = u.activeReservation + r
}
run makeReservation for 3

pred endReservation(u, u': User, r: Reservation, i: Invoice) {
    u'.activeReservation = u.activeReservation - r
    u'.invoices = u.invoices + i
}
run endReservation for 3

pred startRide(u, u': User, r: Reservation, ri: Ride) {
    u'.activeReservation.ride = u.activeReservation.ride + ri
}
run startRide for 3

pred endRide(u, u': User, r: Reservation, ri: Ride, i: Invoice) {
    u'.activeReservation.ride = u.activeReservation.ride - ri
    endReservation[u, u', r, i]
}
run endRide for 3

pred reportMinorIssue(c, c': Car, m: MinorIssue) {
    c'.cumulativeNotifications = c.cumulativeNotifications + m
}
run reportMinorIssue for 3

pred applyBFee(i, i': Invoice, b: BFee) {
    i'.fees = i.fees + b
}
run applyBFee for 3

pred applyADiscount(i, i': Invoice, a: ADiscount) {
    i'.discounts = i.discounts + a
}
run applyADiscount for 3

pred applyCDiscount(i, i': Invoice, c: CDiscount) {
    i'.discounts = i.discounts + c
}
run applyCDiscount for 3

pred showNoNotification {
    #Ride > 0
    #Notification = 0
    #Operator = 0
    #User > 0
    #Car > 0
    #SafeParkingArea > 0
    #SpecialParkingArea > 0
}
run showNoNotification for 3

pred showNoReservation {
    #Reservation = 0
    #Notification > 0
    #Operator > 0
    #User > 0
    #Car > 0
    #SafeParkingArea > 0
    #SpecialParkingArea > 0
}
run showNoReservation for 3
```

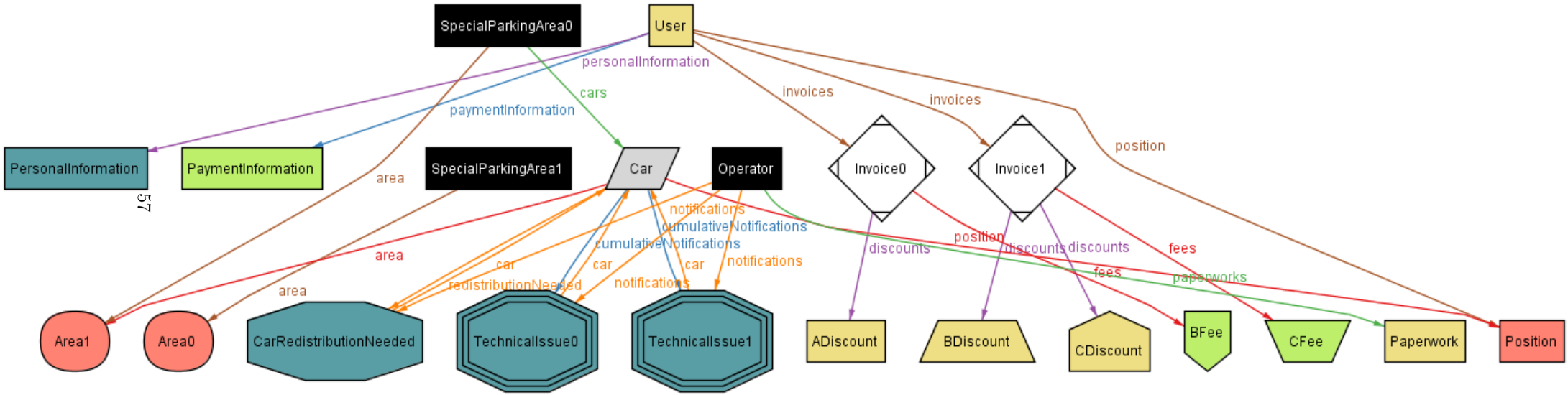
5.5 Results

```
18 commands were executed. The results are:
   #1: No counterexample found. makingReservation may be valid.
   #2: No counterexample found. endingReservation may be valid.
   #3: No counterexample found. startingRide may be valid.
   #4: No counterexample found. endingRide may be valid.
   #5: No counterexample found. reportingMinorIssue may be valid.
   #6: No counterexample found. applyingBFee may be valid.
   #7: No counterexample found. applyingADiscount may be valid.
   #8: No counterexample found. applyingCDiscount may be valid.
   #9: Instance found. makeReservation is consistent.
   #10: Instance found. endReservation is consistent.
   #11: Instance found. startRide is consistent.
   #12: Instance found. endRide is consistent.
   #13: Instance found. reportMinorIssue is consistent.
   #14: Instance found. applyBFee is consistent.
   #15: Instance found. applyADiscount is consistent.
   #16: Instance found. applyCDiscount is consistent.
   #17: Instance found. showNoNotification is consistent.
   #18: Instance found. showNoReservation is consistent.
```

6 Generated worlds

In the following diagram we represent the general world generated through Alloy Analyzer, in order to show the consistency of our alloy model. To make the representation readable, in the first model we considered a world where no notifications and no operators are present, while in the second model we considered a world where no reservations are present. Both cases are produced by the predicate show() for 3 cases. We decided not to attach all diagram not to increase the complexity to the point of making the document unreadable.


User3
User2
PersonalInformation0
PersonalInformation1
PaymentInformation0
PaymentInformation1
User1
SpecialParkingArea0
SpecialParkingArea1
Reservation0
Reservation1
PersonalInformation2
PaymentInformation2
Invoice0
SpecialParkingArea2
Invoice1
Invoice3
Invoice2
Car0
Car1
Ride
User0
Area2
Area1
Area0
ADiscount
BFee2
BFee1
CFee
BFee0
1
PaymentInformation3
PersonalInformation3
Position1
Position0
personalInformation
paymentInformation
activeReservation
position
invoices
cars
car
ride
area
fees
discounts
passengersNumber


SpecialParkingArea0
User
personalInformation
cars
paymentInformation
invoices
invoices
position
PersonalInformation
PaymentInformation
area
SpecialParkingArea1
Car
Operator
Invoice0
Invoice1
notifications
cumulativeNotifications
cumulativeNotifications
position
discounts
discounts
discounts
fees
fees
paperworks
area
area
car
redistributionNeeded
car
notifications
car
notifications
Area1
Area0
CarRedistributionNeeded
TechnicalIssue0
TechnicalIssue1
ADiscount
BDiscount
CDiscount
BFee
CFee
Paperwork
Position

7 Appendix

7.1 Used tools

The following software tools were used to produce the RASD document:

1. TexWorks (www.tug.org/texworks/): framework for LaTex text format
2. AlloyAnalyzer (www.alloy.mit.edu): alloy modeling software
3. Microsoft Visio (www.products.office.com/it-it/visio/flowchart-software): Microsoft tool for modeling charts and diagrams
4. Signavio (www.signavio.com): online framework for modeling class diagrams
5. Notepad++ (notepad-plus-plus.org): alloy output visualization software

7.2 Hours of work

The team divided the work into equivalent parts, even when modeling different parts of the document. In the following table we present a resume of the work division.

Lo Bianco Riccardo	61h
Manzoni Mirco	70h
Mascellaro Giuseppe	73h