Requirement Analysis and Specification Document

Patricia Abbud
Maddalena Andreoli Andreoni
Paolo Cudrano

Contents

1 Introduction

1.1 Purpose

The document you're approaching to read is the *Requirement Analysis and Specification Document* (from now on RASD) for the information system *PowerEnJoy*. The purpose of this document is to describe, with varying degrees of depth and detail, the system we're going to implement. The system will be described firstly by listing the needs of our stakeholders (Goals section). From these goals we'll derive the functional and nonfunctional requirements (requirements section) needed to describe the system, and we'll also underline the constraints (constraints section) and limits of the software given by the world in which it operates (domain properties section). We'll then proceed to describe a series of scenarios and use cases that will probably occur after deployment.

1.2 Actual system

The team has been asked to develop the system for a car sharing service. We suppose that nothing has been create until now and that we do not have to modify or expand any pre-existing application. We will have instead to create the entire system.

1.3 Scope

The system we are going to develop is a car–sharing service called *PowerEnJoy*, based on mobile application and which has a website. The main functionality of the system will be to allow its users to locate and reserve a car to drive in the municipality of Milan and its surroundings using the mobile application. The users may choose to either search for cars in their whereabouts or near a given address, and can drive it inside the geographical boundaries set by the company. Furthermore, the system will be developed such that the users will be encouraged in their good behaviours with discounts and sanctions to the fare per minute. Naturally, the system must also allow the registration of new users; for the registration the system requests both personal and payment information. Then, to complete registration, any guest must send a pdf with driving license and personal ID to the company's email to be validated. All back-end operations are managed by one or more system administrators, who can dispatch on-site operators for emergencies, validate registrations, ban users. The administrators also have the possibility to choose some of the parameters of the system.

1.4 Goals

1. The system allows guests to register; to complete the registration procedure the system sends a password to the guest as an access key.

2. The system should enable a registered user to find the location of an available car within a certain distance from the user's location or from a specified address.

3. The system enables user to reserve a single available car in a certain geographical region for one hour before the user picks it up. If the car is not picked up by that

time, the reservation expires, the system tags this car as available again and it charges the user a fine of 1 EUR.

4. The system should allow the user to employ a car in a proper and safe way.
5. The system charges the user for a predefined amount of money per minute. A screen on the car notifies the user of the current charges.
6. The system starts charging the user as soon as the car ignites. It stops charging them when the car is parked in a safe area and the user exits the car.
7. The system should encourage good user behaviour through the application of discounts to the fee per minute.
8. The system should discourage bad behaviour through the application of sanctions to the fee per minute.
9. The system should provide an alternative usage mode for cars called *money saving option*. Besides aiding the user in saving money, this mode allows for a uniform distribution of cars throughout the city by suggesting the user where to park.
10. The system allows the company to assist the users in case of need and take care of the cars.
11. The admin should be able to configure some parameters of the system.

1.5 Domain properties

We assume that the following properties hold in the analyzed world:

- A tablet is permanently installed in every car.
- The tablet is alimented through the car and cannot be switched off.
- All cars are equipped and located with a GPS system provided by the tablet.
- All GPS systems always give the right position.
- GPS tracking is always on.
- All cars has sensors to detect the presence of passengers in each seat.
- All cars are equipped with a Bluetooth system provided by the tablet.
- The Bluetooth system is always on.
- All cars are equipped with an integrated alarm as theft deterrent.
- All cars are equipped with an accident detection system which is always able to notify the company of an accident when the driving user can't.
- A user can provide their location with their phone's GPS whenever they want.

- The safe areas are predefined and within the municipality of Milan.
- The payments of all services are managed by an external company, which guarantees fulfillment.
- The cars always ignite when they have more than 3% of power charge.
- There is no policy for time limits for a user to retain and use a reserved car.
- To complete the registration procedure, the user must send their driving license and ID as pdf documents to the company email address.
- The company accepts only european licenses as valid.
- Localization services and maps are provided by an external company, which guarantees precision.

1.6 Glossary

User We will refer to all people who are registered to the system as 'users'. All users have personal profiles which contain the following information:

- First name;
- Last name;
- Email;
- Telephone number;
- Username;
- Password;
- Payment information; this in particular includes:
  - Credit card owner;
  - Credit card number;
  - Credit card expiration date;
  - CVV number.
- Driving license information; this in particular includes:
  - License number;
  - Issued date;
  - Expiration date;

And, optionally:

- Personal photo;

Guest We shall call 'guests' all people who are using the interface of the system without being registered or logged in. Guests can't access any functionality of *PowerEnJoy* except for the registration process and the log in.

Parking areas Also called *Safe areas*, parking areas are predefined parking slots within the municipality that are reserved for the car–sharing system *PowerEnJoy*.

Special parking areas , or *recharging areas* are *Parking areas* that provides a plug to recharge the car; in recharging areas, for every parking slots there is a plug connected to the power grid. Recharging parking areas are a subset of Parking areas (parking areas *may* be recharging areas, while the contrary doesn't apply).

Reserved car We will call 'reserved car' a specific car that is booked by a user, who still has to reach and open it, and not taken by other users.

Available car An avalable car is a car that is not reserved by any user.

Power grid We shall call power grid the system of electrical distribution that covers all recharging areas and is privately owned by the company *PowerEnJoy*. The thus defined power grid is linked with the public power grid from which it takes the needed electricity.

Standard price We shall call 'standard price' the price per minute charged to the user, without any discount or sanction applied.

Discount A discount always lowers the price per minute charged to a user. It is a negative percentage that is applied every time a user has a virtuous behaviour.

Sanction A sanction always increases the price per minute charged to a user. It is a positive percentage that is applied every time a user has a wasteful or incorrect behaviour.

Money saving option An option offered to the users by the system. The user inputs their final destination and the system indicates them the best close station where to leave the car.

Time Window A time window is a period of time allotted after every ride when the user can still operate with the car, i.e. open it and plug it to the power grid, even if the car is no longer reserved to them. The charges are invoiced after the time window finishes. The time window finished either after a fixed amount of time or when another user reserves the car before then.

Standard ride We define as *standard ride* every ride that finishes with the car being parked in a safe area and where no emergency has occurred.

Uniform distribution We shall call distribution the number of cars parked at a given point in time in all the parking areas of *PowerEnJoy*. An uniform distribution is a particular distribution calculated through an algorithm provided by the customers, that satisfies a series of constraints.

1.7 Assumptions

The assignment document was unclear and ambiguous on some points of the specifications. Hence, we will make the following assumptions:

- Parking areas and special parking areas are two different things; however, common sense suggests that it isn't logical to charge users if they plug the car to the power grid in a recharging area but are not parked in a safe area. Neither it makes sense to sanction them if they park it in a safe area that is 3 km far from the power grid. Having the two areas separate would lead to the consequences above, so we decided that while a safe area may not be a recharging area, recharging areas are always safe areas. Furthermore, assuming that a city has many safe areas, so that users can enjoy the service all around the city, it is reasonable to think that just a few of them have been equipped with a power grid connection, for economic and infrastructural reasons.

- The users are able to reserve an available car only from their geographical region.

- There is a "manual" way to close and open the car, i.e. that the user is allowed to temporarily park the car – while still being charged –, get out, close the car, and then get back and open it again. This means that the system can be used not only for one-way travels, but also when more stops or a round trip is needed.

- Parking areas and special parking areas are allotted and private parking spaces owned by *PowerEnJoy* and distributed throughout the urban area.

- We assume that all cars are the same model and have the same features; specifically, they are all 5-door Citroën C-Zero Micro Car with four seats, customized for the purposes of *PowerEnJoy*.

- If there is at least one passenger, the system cannot infer if the driver is actually the user or the user is the passenger and someone else is driving. There is no way of knowing that; however, we assume that upon registration the user has accepted the policy that asks them to be the only driver. If they do not comply to that and commit some infractions, the company reserves the right to take legal action.

- The text of the assignment does not say how parking areas are selected when the user has chosen the money saving option. We assume that our clients has given or will give us a computable and feasible algorithm to manage priorities in suggestions and to find the "uniform distribution" of cars also based on power plug availability.

1.8 Constrains

1.8.1 Regulatory policies

According to privacy law the system must require the User to give permission before getting his position and before acquiring, storing and processing his personal data.

1.8.2 Hardware limitations

- Mobile application
  - Space for app package
  - 3G connection
  - GPS
- Website
  - Modern browser: the Website of the service must work with all versions of Google Chrome, Opera, Firefox and Internet Explorer released after 2010.

1.8.3 Interfaces to other applications

PowerEnJoy has to interface with the payment service application in order to invoice users for the rides.

1.9 Identifying stakeholders

Main stakeholders are the inhabitants of Milan willing to use an ecologic and comfortable mean of transport, without the need to buy and maintain a personal car. The system can be adopted in any other city, so the inhabitants of other cities too can become stakeholders. In Milan, the project is promoted and participated my the city council as minor investor.

2 Actors identifying

The system involves as main external actors:

User The final target of the system. Once registered, they can reserve a car and use it in respect of the agreements established with *PowerEnJoy*. They are charged for the usage of the car.

Guest A potential user not registered in the system yet.

The following actors have also been identified inside *PowerEnJoy* as needed by the system:

Admin Back-office administrators, they have access to a control panel.

Operator Maintenance operators who provide field-based assistance and ensures cars are always ready to be used.

The system has moreover to interact with the following external service provider:

Payment service Provider of every service related to users payment management.

Location service Provider of services related to the localization and map services.

3 Requirements

3.1 Functional requirements

We assume that all domain properties stipulated in paragraph 1.6 hold. We decided to use a *goal-driven* method to structure our requirements, meaning that we decomposed the high-level goals of paragraph 1.5 into low-level requirements. Some of these requirements stem directly from the requests of our clients, while others are born from the necessity to have a sound system.

For each goal, we derive the following requirements:

- *G[1]* The system allows guests to register; to complete the registration procedure the system sends a password to the guest as an access key.
  - R[1.1] The system has to allow any person to submit only one account request.
  - R[1.2] The system must accept an account requrest only if the credit card owner's name and the user's name coincide.
  - R[1.3] The account is created when an admin validates all the necessary data.
  - R[1.4] The system must be able to generate passwords.
  - R[1.5] The system has to send a newly generated password to the user via email when the account is created.
  - R[1.6] The system must be able to check whether a password is correct or not.
  - R[1.7] The system must let the user log in only if the password is correct.
  - R[1.8] The system has to generate a new password and send it via email if the user asks for it.
- *G[2]* The system should enable a registered user to find the location of an available car within a certain distance from the user's location or from a specified address.
  - R[2.1] The system must have the ability to locate the user if the user's GPS is on.
  - R[2.2] The system is able to locate any valid address.
  - R[2.3] The system is able to find any of the parking areas of the company.
  - R[2.4] The system is able to identify available cars inside parking areas.
  - R[2.5] The system lets users see whether there are available cars in a specified radius.
- *G[3]* The system enables user to reserve a single available car in a certain geographical region for one hour before the user picks it up. If the car is not picked up by that time, the reservation expires, the system tags this car as available again and it charges the user a fine of 1 EUR.
  - R[3.1] The system allows users to reserve an available car.
  - R[3.2] The cars cannot be reserved by more than one user at any given time.

  - R[3.3] The system keeps the current reservation standing until the user has opened the car or an hour has passed.
  - R[3.4] The system autonomously cancels a reservation if the user who has reserved it hasn't picked it up after one hour.
  - R[3.5] The system must impede any user with an expired license to reserve a car.
  - R[3.6] The system must impede any banned user to reserve a car.

- *G[4]* The system should allow the user to employ a car in a proper and safe way.
  - R[4.1] The system must be able to locate any car at any given time.
  - R[4.2] The system detects whether there are passengers inside a car, and how many.
  - R[4.3] The system must be able to collect data about the power charge of any of its cars.
  - R[4.4] The system detects when a severe accident has occurred to a car.
  - R[4.5] The system must be able to detect when a user is near a car.
  - R[4.6] The system must be able to tell when a car is parked in a safe area.
  - R[4.7] The system must be able to detect when a car is plugged to the power grid.
  - R[4.8] The system must be able to detect whether the driver is still in the car.
  - R[4.9] The system automatically unlocks a car when the user that has reserved is nearby.
  - R[4.10] The system automatically locks a car when the user has exited it inside a safe area.
  - R[4.11] The system allows the user to lock and unlock their car manually when outside a safe area.
  - R[4.12] The system provides a finite time window that begins when the user exits the car inside a safe area. The time window must either end when the allotted time is finished or when another user reserves the same car.
  - R[4.13] The system allows the user to re-enter the car if the time window is still open and the user tries to manually open it.

- *G[5]* The system charges the user for a predefined amount of money per minute. A screen on the car notifies the user of the current charges.
  - R[5.1] The system is able to retrieve all data necessary to charge the user. This data is the duration of the ride and all the conditions for the eventual application of discounts and sanctions.
  - R[5.2] The system notifies the user of the fee per minute he's being charged through a screen inside the car.

  - R[5.3] The system notifies the final charges to the user after the time window for the ride has expired.
  - R[5.4] The system invoices the user after the time window for the ride by communicating the charges to the external payment system.

- *G[6]* The system starts charging the user as soon as the car ignites. It stops charging them when the car is parked in a safe area and the user exits the car.
  - R[6.1] The system charges the user with a fee per minute.
  - R[6.2] The system can tell when the car has ignited.
  - R[6.3] The system stops counting the charges for a standard ride when the user exits the car while inside a safe area.
  - R[6.4] The system starts charging the user when the car ignites or when the reservation expires while the user is inside the car.

- *G[7]* The system should encourage good user behaviour through the application of discounts to the fee per minute.
  - R[7.1] The system applies all discounts to the fee per minute.
  - R[7.2] If the conditions for a discount are satisfied only for a limited period of time during the ride, the discount is applied only to that time period.
  - R[7.3] The system applies a discount every time a user brings two or more passengers in the car with them and the car is on.
  - R[7.4] The system applies a discount to the total fee for the ride when the car has more than 50% of power charge by the end of a standard ride.
  - R[7.5] The system applies a discount to the total fee for the ride every time the car for that ride is plugged to the power grid at the end of the time window.
  - R[7.6] The system calculates the total discount as the sum of the singular discounts applied to the ride. If this sum exceeds 100%, then it is considered simply as 100%.

- *G[8]* The system should discourage bad behaviour through the application of sanctions to the fee per minute.
  - R[8.1] The system applies a sanction every time a car is returned in a safe area at more than 3 km from the nearest recharging safe area.
  - R[8.2] The system applies a sanction every time a car is returned with less than 20% of power charge.

- *G[9]* The system provides an alternative usage mode for cars called *money saving option*. Besides aiding the user in saving money, this mode allows for a uniform distribution of cars throughout the city by suggesting the user where to park.
  - R[9.1] The system allows a user to select the *money saving option* at any point during their ride.

- R[9.2] The system applies a discount every time the car is returned in the suggested safe area with the *money saving* option.
- R[9.3] The system selects the safe area suggested to the user through the available algorithm. (see text assumptions)

- *G[10]* The system allows the company to assist the users in case of need and take care of the cars.
  - R[10.1] The system allows a user to notify the back-end administrators if they have a problem with the car.
  - R[10.2] The user must specify whether this problem is an accident or a reparation.
  - R[10.3] The system can locate *PowerEnJoy* operators.
  - R[10.4] The system allows administrators to know the status of on-site operators and dispatch them if they are available.
  - R[10.5] The system keeps track of every car's battery and alerts the admin when it lowers below 3%.
  - R[10.6] The system notifies the admin when it detects that an accident occurred.
  - R[10.7] If the system has already notified the admin of an accident, then it must prevent the user from notifying it again.
  - R[10.8] If the system has already notified the admin of an accident, then it must notify the user of this fact.
  - R[10.9] Every time the system detects or a user notifies a problem, an emergency report is opened.
  - R[10.10] Operators can change an emergency report status only when they are assigned to it.
  - R[10.11] The system considers all type of emergency reports related to a reparation as on-site at first.
  - R[10.12] The system allows operators to change the type of emergency reports related to a reparation from on-site to not on-site.
  - R[10.13] Operators can change the type of emergency report from accident to reparation or vice versa only if they are assigned to it.

- *G[11]* The admin should be able to configure some parameters of the system.
  - R[11.1] The system allows the aministrators to modify parameters that influence the usage of cars. This parameters are:
    1. The standard fee per minute
    2. The lost reservation fee (starting value = 1EUR)
    3. The time before considering a reservation lost (starting value = 1 hour)
    4. The fine for leaving an accident site

5. Discount percent values:
    * Passengers discount (starting value = 10%)
    * Residual battery discount (starting value = 20%)
    * Plugged car (starting value = 30%)
    * Money saving option (starting value not defined)
6. Sanction percent values:
    * Residual battery sanction (starting value = 30%)
    * Distance from power grid sanction (starting value = 30%)
7. Time window for reservation fee
8. Time window after ending the ride for reopening and pluggin the car.
9. The fine for unnecessary emergency request.

3.2 Non-functional requirements

Even thought our clients did not explicitly state any non-functional requirement for the system to be implemented, after some consideration we thought that the following requirements should be included:

Integrity Data in the system must not be exposed to accidental or malicious modification or destruction.

Confidentiality The system must be so it is difficult to breach the privacy of all users, above all because of the sensitive information we store (i.e. credit card information).

Robustness We want the system to be able to process and cope with wrong input from the users. In particular, this property must hold for the system's relationship with the localization services provider.

Performance The system should ensure a short response time for all computations, and must be able to sustain a high number of simultaneous requests.

Flexibility The system must be flexible to expansions in functionalities, i.e. it must be possible to easily integrate new functionalities in the system without losing in robustness and reliability.

Availability The system must provide an availability of at least 99.9% for what concerns the most sensible components, and must prevent car theft or damages to the users during the downtimes.

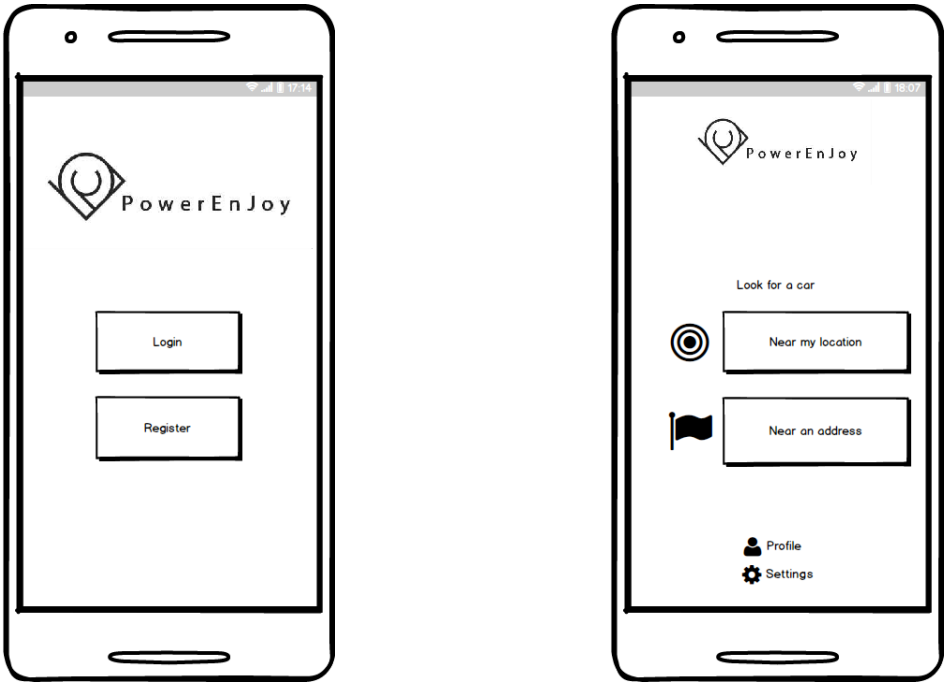
3.2.1 User interface

The interface of our application is thought to be used mainly via mobile app, with some content also displayed in a web page. The team reasoned that the main functionalities of the system (such as reserving and managing a car) make sense only in a *movable* context (meaning, such that the user can do them anywhere they have an internet connection). Those functionalities are available on the mobile app. On the other hand, operations such

as signing up are more easily managed in front of a computer, so the web page allows users to register, login and manage their profiles (which they can do also via mobile app anyway).

We pondered on the possibility of adding the possibility of reserving a car via web browser. However, upon consideration, we decided it is not a good mechanism: with that functionality, a user could very well be registered without having downloaded the app and reserving a car without being able to access it in any way. Hence, the reservation of *PowerEnJoy* cars can be made only via mobile app.

Mobile app



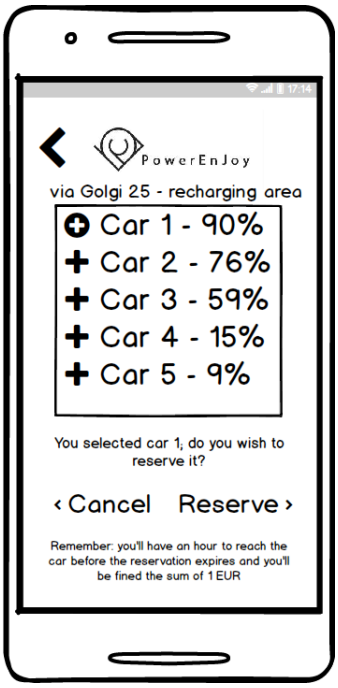
(a) Guest view (b) User view

Figure 1: Mobile app: home page view

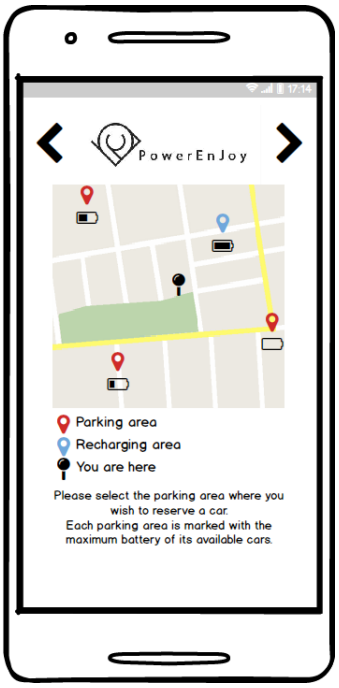
Figure 1 shows the first page that is shown when entering the app. Picture 1.a is the guest view, who has only the possibility of either logging in or registering in the system. Picture 1.b shows the user view. The user has more functionalities: they can look for cars in their vicinity or near an address, see and edit their profile, and changing settings (notification and sound settings).

Figure 2 shows the main functionalities of the *PowerEnJoy*'s app: reserving a car (2.a and 2.b) and what you can do while using a car (2.c).

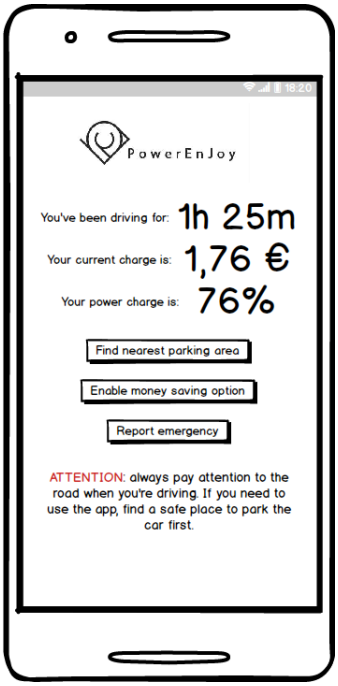
Figure 3.a and 3.b shows, respectively, the website homepage from the standpoint of a guest and the sign up page. We haven't drawn the website mockup from the standpoint of a registered user since there are no more functionalities than those already present in the mobile app, as said above.



(a) View for the reservation



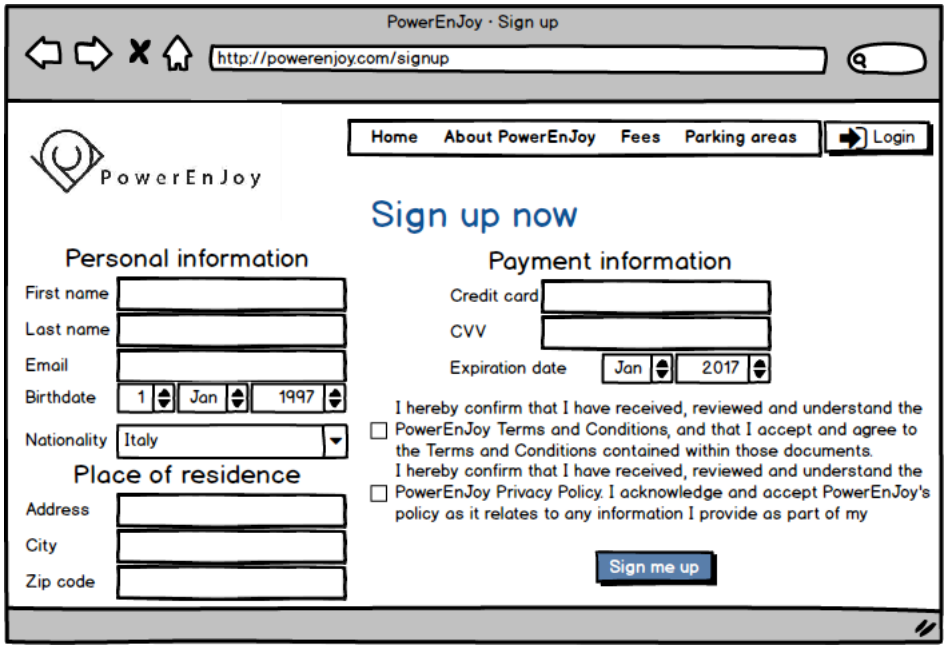
(b) View after having selected a parking area



(c) Display of the app while using a car

Figure 2: Mobile app: main functionalities

(a) Website homepage



(b) Sign up on website

Figure 3: Website

Documentation

In order to keep track of all phases of the development process and the overall structure of the system, the team will release the following documents:

RASD , Requirement Analysis and Specification Document, which provides a thorough description of the system, the requirements and the specification thanks to the use of UML models.

DD , Design Document, which contains a more in-depth description of the functionalities of the system.

ITPD , Integration Test Plan Document, which describes integration tests and the team's intended plan to accomplish them.

PP , Project Plan, which defines a planning for the development project.

4 Scenario identifying

4.1 Scenario 1: Registration from the website

Anakin has just moved to Milan and has rented a flat; however, he couldn't afford a place close to the city centre, where he works; he also doesn't have a car, so he's been going back and forth with public transport. Because of that, he needs to wake up half an hour earlier and usually gets home very late, and he's getting tired. He then decides to look for a solution on Google, and he finds the car–sharing service of *PowerEnJoy*, which has a parking place close to his home. The *PowerEnJoy* web page has all the information readily available, pricing, features, an approximated map of the parking areas included and a link to download the application from suitable store, so Anakin decides to sign up. He completes a form, where he writes his complete name, personal information, credentials and payment; then he scans his driving license and personal ID and sends them via email to *PowerEnJoy*. The system admin checks the license with the local DMV and approves Anakin's request. The system then sends him an email with the password he will need to access the system.

4.2 Scenario 2: Login in the app

Padmé has registered on the *PowerEnJoy* website and now has downloaded the application on her smartphone. Opening the application, she finds a screen asking her to log in. Padmé registered with *Naboo_princess* as username, and the password she received in her email is *7aKmm93s*, so she fills the login form with this information. The first time she writes the password wrong, so the login is rejected and the application asks her to try again. The second time she types the password right, so the system accepts the login and shows the main page of the application.

4.3 Scenario 3: Reserving and using a car

Luke must reach his aunt and uncle for the usual sunday roast. He doesn't have a car, and usually he just takes the subway. However today the public transport workers are on strike, and the metro is out of order. Luke is a distracted kid, always with his head in the clouds, so he forgot about the strike and has walked for the fifteen minutes needed to reach the metro. He is nevertheless smart and resourceful, and so he remembers that he's signed in the *PowerEnJoy* system. He opens the app and presses the "look for a car near my location" button. He has the GPS activated, so the system locates him and tells him that there's a parking area with an available car next to the metro station. The application gives him the choice of reserving the car or cancel the operation, and Luke reserves the car.

4.4 Scenario 4: Parking and regular fees

Leia needs to get to the american consulate ASAP. She's a frequent user of the car–sharing service, so she already knows all the safe areas where she can park around the diplomatic block in the city centre, since she often needs to go there. While she drives, the smart

display in her car tells her how much the system is currently charging: the fee per minute is 0,50 EUR, and she's been driving for half an hour, so her fee currently is 15 EUR. She reaches the diplomatic area after two more minutes, and she knows that the parking area is around the corner from the consulate, so she reaches there. This particular safe area is not a recharging area, and Leia left the car with a 40% battery full, so the system charges her 16 EUR. The display notifies her that she's parked in a safe area and that no sanction or discount applies to her fee. She exits the car and the system locks it automatically.

4.5 Scenario 5: Power grid and discounted fees

Boba Fett has arrived to a recharging area and parks there. He notices that the battery is under 20%, so he decides to plug the car in the power grid. When he exits the car, the system closes it automatically and notifies him of the time window where he can plug it or reopen it. He plugs the car to the power grid while the time window is still open and then goes his way. After the time window closes, the system detects the car is plugged in and notifies Boba Fett of the final charges, discount for pluggin the car included.

4.6 Scenario 6: Parking outside safe areas

Han, Leia's husband, is also a regular user of the system. He's however less abiding to rules and is a bit of a free spirit, so he generally never gains any discount and is often sanctioned for wasteful behaviour. For example, last monday he needed to reach his bank in Porta Genova for a meeting with his broker. He was very late, so he parked right outside the bank, outside any safe area. It took him twenty minutes to get there, so the system had charged him 10 EUR. As soon as he stops the car, the display notifies him that he is in an unsafe area, so he'll keep being charged even if he isn't using the car. Han confirms and exits. The car doesn't lock automatically, so he needs to turn his bluetooth on and close it with the *PowerEnJoy* application.

4.7 Scenario 7: Lost reservation

Han and Leia want to go a restaurant. Han thinks that they are about to leave the house, so he decides to reserve a car while still at home to make sure they will have a car after leaving home, but it takes Leia more than an hour to get prepared. In one hour they are not near the car to open it, so the system cancels Han's reservation, notifies him about the 1 EUR sanction charged for uselessly reserving it and asks if he wants to reserve another available car.

4.8 Scenario 8: Two passengers + special parking area

Chewbacca met Han and Leia on his way, so he invited them to take one car. When they sit in the car, Chewbacca receives a notification on the display that two other passengers are detected. When they arrive to their destination, Chewbacca sees on the application's map that there is a special parking area near them, therefore he decides to put the car there and doesn't forget to plug in the car to the power grid. At the end of the ride

Chewbacca gets a discount of 40% (10% for the 2 passengers + 30% for leaving the car on the special parking area and plugging it to the power grid).

4.9 Scenario 9: Money saving option

Lando Calrissian has decided to go to the movies tonight: a new Star Wars movie is out and he heard it's very good. He is not in a hurry, so he reserves a car and decides, once he gets there, that he might as well turn the money saving option on to receive a discount. He inserts his destination in the system and presses enter; the system tells him that he'll need to park the car in a recharging area 0.5 kilometres from the cinema. Lando starts driving and arrives at the recharging area. He then exits the car and the system closes it automatically. He doesn't plug the car in and after the time window (in which he's already started walking towards the cinema) the system tells him the final income, discount for using the money saving option included.

4.10 Scenario 10: Recover a car with user input

Obi-Wan is driving peacefully to reach his yoga instructor, Qui-Gon Jinn, when the car breaks down; he tries to restart it or check what is wrong, but the system doesn't detect any anomaly and he isn't a mechanic, so he decides to notify the company: he opens the app on his phone and enters the *Emergency help* section. The app asks him to broadly describe the problem by filling a form. He then selects the option "car doesn't work" and the system tells him that an operator is on his way. In the meantime the system, having located the car, notifies the admin of its whereabouts. The admin checks which operator is closest to the site and available and sends him. The operator – which is also a mechanic – checks the car on site and then, since the car needs more serious repairs, he notifies the admin that the reparation cannot be carried out on-site. While the admin dispatches another operator with a tow truck, the system informs Obi-Wan of the final charges (calculated until the notification of the breakdown). The car is no longer his. The new operator arrives and tows the car to the company's garage. She contacts the insurance company, that comes and checks the car. The insurance decides the breakdown was not Obi-Wan's fault, and pays the cost of the repairs.

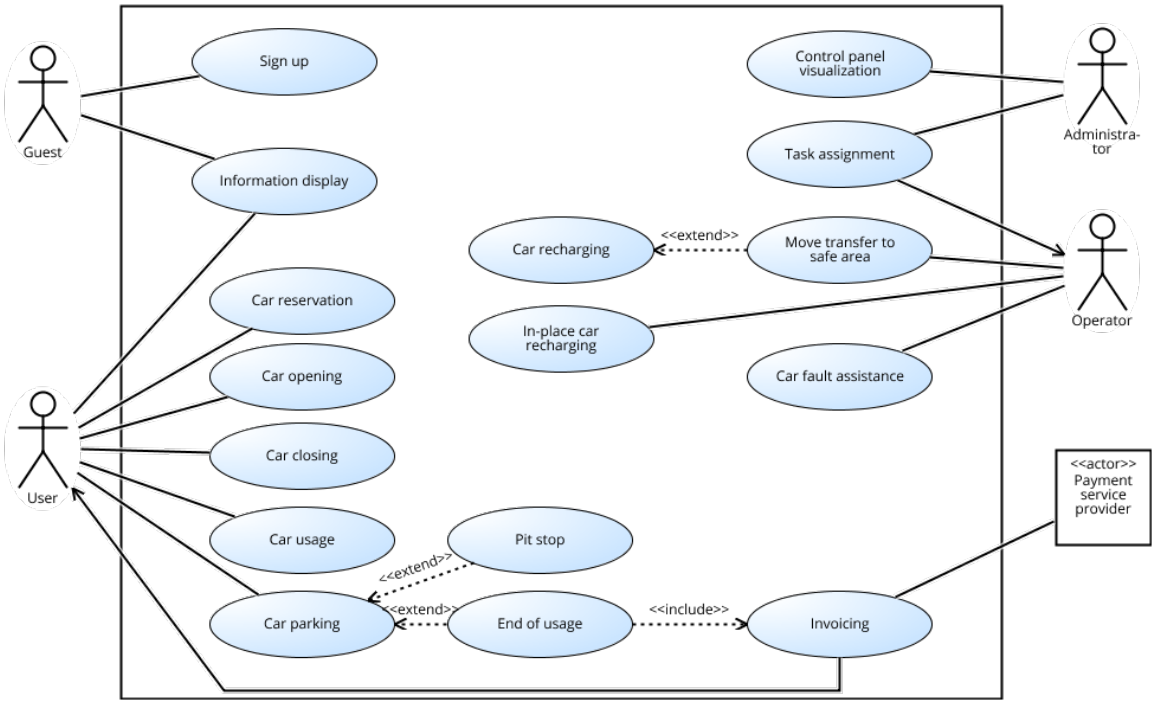
4.11 Scenario 11: Manually assist parked car

An unnamed user has left a car with ¡3% battery in a safe area far from the power grid. The system locates the car and alerts an admin of the situation. The admin locates the operator with a tow truck closest to the site and assigns the maintenance to him. The operator was free so he accepts and reaches the car. He tows it to the nearest recharging area and plugs it there. The operator then notifies the system that he completed the maintenance operation.

5 UML models

5.1 Use case diagram

A global picture of the system interaction with actors is provided here by means of use case diagrams. Following, an analysis of the most interesting use case situations derived from scenarios is presented.



5.1.1 Use case 1: Register in the system

Name Register in the system

Actors

Guest The guest who wants to register in the system.

Admin The system administrator.

Entry condition The guest decide to register to the system from the website or from the app.

Flow of events

1. The guest fills the form with their personal information.
2. The guest fills the form with their payment information.
3. The guest reads and accepts privacy policies and agreements from the company.
4. The guest sends the registration requests.

5. The system notifies them that to complete the request the guest will need to send an email with personal ID and driving license in pdf attached.
6. The guest sends the necessary documents.
7. The admin receives the email and checks with the DMV (Department of Motor Vehicles) the correctness of the information.
8. The admin accepts the registration request.
9. The system sends the user via email a password to enter the system.

Exit condition The guest has a new profile and has become a user. They can now reserve car and use all functionalities of the system.

Exceptions

- **The credit card owner's name doesn't coincide with the guest's name.** The system trivially checks for equality of the two fields. If they are different, it notifies the guest of the discrepancy and the company policy (i.e. that the payment method must belong to the user themselves and not to third parties). The operation is aborted.
- **The license is not valid**. The admin notifies user via email. The registration request is cancelled.

5.1.2 Use case 2: Reserve a car

Name Reserve a car

Actors

User The user who wants to reserve a car.

Entry condition The user decides to reserve a car to take within the next hour.

Flow of events

1. The user logs in into the mobile app and goes to the reservation section.
2. The system automatically retrieves and displays the location of the user, but the user can specify a different location if needed.
3. The system displays the position of the available cars close to the selected location.
4. The user selects a car and confirms the reservation.

Exit condition The system reserves the car for the user.

Exceptions

- **The system is not able to find a position inserted manually.** The user is informed and the operation is aborted.

- **There are no available cars.** The user is informed and the operation is aborted.
- **The user cancels the operation before confirming.** The reservation process is not completed and the car remains available to other users.
- **The user doesn't come get the car before one hour.** The user is notified of the lost reservation. The system charges them the lost reservation fee. The car becomes available again.

Special Requirements None.

5.1.3 Use case 3: Park in known safe area

Name Park in known safe area

Actors

User The user of the car.

Car The car in use.

Entry condition The user is driving and has reached their destination. They know a safe area close to the destination.

Flow of events

1. The safe area is free and the user parks in it.
2. As the car is turned off, the system detects it is in a safe area.
3. The user exits the car.
4. The time windows starts.
5. The system closes the car.
6. The time window ends.
7. The system charges the user for the ride.

Exit condition The user leaves the car and the car becomes available to other users.

Exceptions

- **The safe area is taken.** The user can't end their ride and this operation is aborted.

Special Requirements None.

5.1.4 Use case 4: Park with money saving option

Name Park with money saving option

Actors

User The user of the car.

Car The car in use.

Entry condition The user selects the *money saving option* at some point of their ride and insert their destination.

Flow of events

1. The system indicates the user the suggested safe area for their destination.
2. The user parks in the suggested safe area.
3. As the car is turned off, the system detects it is in a safe area.
4. The user exits the car.
5. The time window starts.
6. The system closes the car.
7. The time window ends.
8. The system charges the user for the ride. A discount is applied for using the *money saving option*.

Exit condition The user leaves the car and the car becomes available to other users.

Exceptions

- **The suggested safe area becomes taken while the user is driving.** The system selects another safe area and notifies the user of the new suggestion.
- **The user parks in another safe area.** The system notifies the user that they will not receive a discount. If the user decides to end the ride anyway and exits the car, the system charges them without applying the *money saving option* discount.
- **The destination of the user changes.** The user selects a new destination and the system indicates another suggestion.
- **The user disables the *money saving option* while driving.** The suggested safe area stops being displayed and the ride continues as normal.

Special Requirements The *money saving option* must be selected before stopping the car in a parking area.

5.1.5 Use case 5: Park in a recharging area

Name Park in a recharging area

Actors

User The user of the car.

Car The car in use.

Entry condition The user is about to park in a recharging area.

Flow of events

1. The user parks the car in the recharging area.
2. As the car is turned off, the system detects it is in a safe area.
3. The user exits the car.
4. The time window starts.
5. The system closes the car.
6. The user plugs the car into the power grid through the supply point installed in the parking space.
7. The system detects the car is recharging.
8. The time window ends.
9. The system modifies the charge applied to the user for the ride. A 30% discount is applied to promote virtuous behaviors.
10. The system charges the user for the ride.

Exit condition The user leaves the car and the car becomes available to other users.

Exceptions

- **The user does not plug the car into the power grid.** The user is charged as if they parked in a safe area.
- **The user plugs the car into the power grid after the time window has closed.** The user is invoiced the moment the time window closes. The user is charged as if they parked in a safe area.

Special Requirements The user plugs in the car during the time window after the ride. Otherwise the discount is not applied.

5.1.6 Use case 6: Application of several discounts

Name Application of several discounts

Actors

User The user who is driving the car.

Entry condition The user has parked in a safe area. Multiple discounts apply to their ride.

Flow of events
1. The user exits the car.
2. The time window starts.
3. The system closes the car.
4. By the end of the time window, the user has done several things that earn him discounts.
5. The system calculates the sum of all percent values of the appliable discounts.
6. The sum of all discounts is applied to the fee.
7. The user is notified and charged the final fee.

Exit condition The user leaves the car and the car becomes available to other users.

Exceptions
- **The sum of discounts exceeds 100%** The sum is counted as 100% and the user is notified that he doesn't have to pay for the ride.

Special requirements None.

5.1.7 Use case 7: Parking outside safe areas

Name Application of several discounts

Actors

User The user who is driving the car.

Entry condition The car is turned off outside a safe area.

Flow of events
1. The system detects the car is outside a safe area.
2. The system notifies the user that they'll keep being charged until they park inside a safe area.
3. The user exits the car.
4. The user closes the car manually through the mobile application.
5. The user comes back after a while and opens the car manually.
6. The user reignites the car and keeps driving.

Exit condition Eventually the user parks in a safe area. The use case *Park in known safe area* is invoked.

Exceptions None

Special requirements None.

5.1.8 Use case 8: Passengers onboard

Name Passengers onboard

Actors

User The user who is driving the car.

Entry condition The user picks two or more passengers up at any moment during the ride (it may also be at the beginning of it).

Flow of events

1. The system detects there are two (or more) passengers in the car.
2. The system starts charging the user the discounted fee for having passengers inside.
3. The passengers get out of the car at any point during the ride (it may also be at the end of it).
4. The system detects there are less than two passengers in the car (may also be one passenger only).
5. The system reverts to charging the user the standard fee.

Exit condition At the end of the ride, the user will be charged with the fee discounted only for the time the passengers were inside the car.

Exceptions None.

Special requirements None.

5.1.9 Use case 9: Manually assist a parked car

Name Manually assist a car

Actors

Admin The administrator who sends the operator.

Operator The operator sent to recover the car.

Entry condition The car is in a safe area without enough power charge to be used.

Flow of events

1. The administrator is notified by the system that a parked car needs manual assistance.
2. The admin assigns the maintenance work to an operator.
3. The operator accepts the assignment and the admin is notified of it.
4. When available, the operator reaches the car.
5. The car is towed to a recharging area and plugged into the power grid.

6. The operator checks the assignment as completed.

Exit condition The maintenance activity has been performed and the admin is notified of the completion.

Exceptions

- **The operator is not able to perform the maintenance.** The assignment is marked as *not completed*, the cause is inserted into the system. The admin will be notified of it and will either assign the problem to another operator or manage it without the help of the system. The assignment will be anyway closed at the end of this process.

Special Requirements The operator cannot refuse an assignment if he is online and always accept it within a working day. An operator is always online when at work.

5.1.10 Use case 10: Manage infractions

Name Manage infractions

Actors

Admin The system administrator.

User The user responsible for the infraction.

Entry conditions The company receives by mail the notification of the infraction.

Flow of events

1. The company pays the fine for the infraction to the police.
2. The admin logs into the system and inserts the license plate and the time of the infraction to find the responsible user.
3. The system shows the name of the user.
4. The system charges the user with the fine.
5. The system notifies the user of the payment.

Exit conditions The legal procedure has been closed and the user has paid the fine.

Exceptions

- **The infraction causes the user to lose their license** The admin temporarily bans the user. The user won't be able to reserve a car anymore until they send another valid license via email to the admin.

Special requirements None.

5.1.11 Use case 11: Assist user after an accident

Name Assist user after an accident

Actors

User The user driving the car.

Admin The administrator who receives the notification from the system.

Operator The operator sent to assist the user.

Entry condition The user is driving and is involved in an accident.

Flow of events

1. The User notifies the Admin that an accident has happened. In case of serious accidents, it is the Accident Detection System of the car, instead of the User, that notifies the Admin. After the notification, the User waits for an Operator to arrive.
2. The Admin dispatches an Operator with a tow-truck.
3. The Operator arrives on site.
4. The Operator manages the jointly-agreed statement for insurance purposes with other drivers involved and takes care of contacting the insurance company.
5. When everything is done on site, the Operator takes the car to the company garage, to be analyzed by the insurance company if needed.
6. The user leaves the site when the Operator takes the car away.

Exit condition Eventually, the car is repaired and the insurance company emits a result on the accident report and refunds the company for the reparation costs of the car. The car is put back in use (the car is taken to a parking area; it can be reserved again).

Exceptions

- **The User leaves the site before the removal of the car.** The Operator can report it to the system, and the user will be charged of an extra. If the User is penally implied in the accident, the Operator provides their personal details to the law enforcement.
- **The insurance asserts that the accident is fault of the User and doesn't take responsibility for it.** The User is fully charged of the reparation costs.

Special Requirements The Operator cannot refuse to be assigned to an accident if he is online and always accept it within 10 minutes. An Operator is always online when at work. The Accident Detection System always detects an accident when the User is unconscious. If the accident detection system detects an accident and the user is conscious, it is the detection system who alerts the system of the accident. The system then prevents the user to send another notification.

5.1.12 Use case 12: Assist a user on-site after a car breakdown

Name Assist user after a car breakdown

Actors

User The user driving the car.

Admin The administrator who receives the notification from the system.

Operator The operator sent to assist the user.

Entry condition The user is driving and notices a car breakdown. An empty battery while driving is considered as a car breakdown.

Flow of events

1. The User notifies the Admin that a breakdown has happened. After the notification, the User waits for an Operator to arrive. The charges for the User are suspended.
2. The Admin dispatches an Operator.
3. The Operator arrives on site.
4. The Operator repairs the car.
5. Once the reparation has ended, the User is assigned again to the car and the charges start again.

Exit condition The Operator marks the assignment as *completed* and leaves. The User is back on their car.

Exceptions

- **The Operator decides that the reparation can't be done on-site.** The use case *Assist a user with a not on-site reparation* is invoked.
- **The User leaves the site before the reparation ends.** The Operator reports it to the system. At the end of the reparation, the User is not assigned to the car anymore and is invoiced for the ride (until the breakdown occurred). The Operator notifies the Admin that the car must be taken to a parking area and leaves the site. The Admin dispatches an Operator with a tow truck. The new Operator takes the car to the closest available parking area.
- **The Operator asserts that there is no need for intervention and the User is still on site.** The User is fined. The car is assigned to the User again and the charges start again.
- **The Operator asserts that there is no need for intervention and the User has gone from the site.** The User is fined. The Operator notifies the Admin that the car must be taken to a parking area and leaves the site. The Admin dispatches an Operator with a tow truck. The new Operator takes the car to the closest available parking area.

Special Requirements The Operator cannot refuse an assignment if he is online and always accept it within 10 minutes. An Operator is always online when at work.

5.1.13 Use case 13: Assist a user with a not on-site reparation

Name Assist a user with a not on-site reparation

Actors

User The user driving the car.

Admin The administrator who receives the notification from the system.

Operator The operator sent to assist the user.

Entry condition The operator sent to fix a breakdown asserts that the reparation can't be done on-site and notifies the Admin of the change. The notification contains a brief description of the reason.

Flow of events

1. The User is not assigned to the car anymore and is invoiced for the ride (until the breakdown occurred). The car cannot be reserved anymore.
2. The Admin dispatches a new Operator with a tow truck.
3. The new Operator contacts the insurance company and notifies the breakdown.
4. The new Operator takes the car to the company garage.

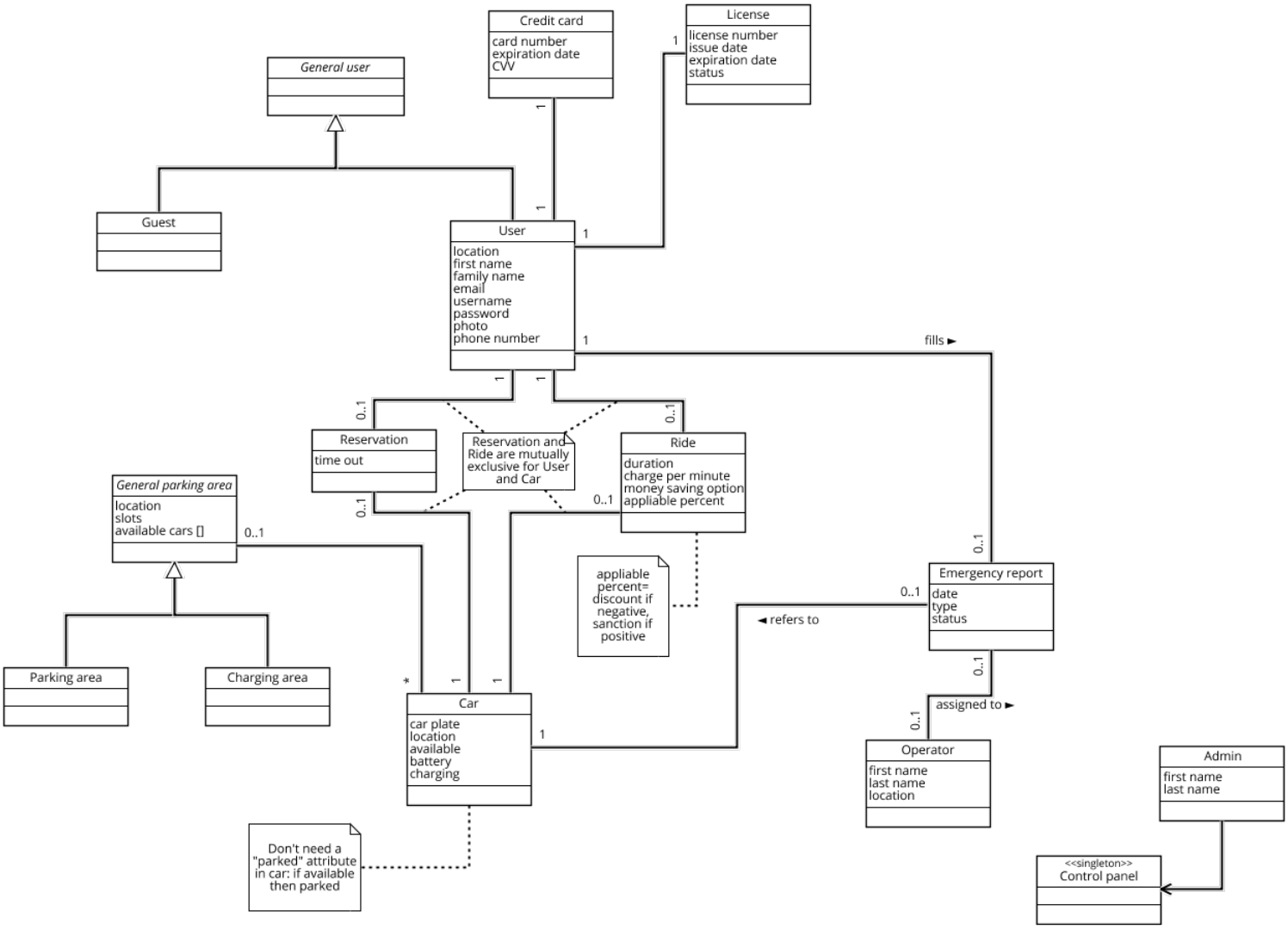
Exit condition Eventually, the car is repaired and the insurance company emits a result on the accident report and refunds the company for the reparation costs of the car. The car is put back in use (the car is taken to a parking area; it can be reserved again).

Exceptions

- **The insurance asserts that the accident is fault of the User and doesn't take responsibility for it.** The User is fully charged of the reparation costs.

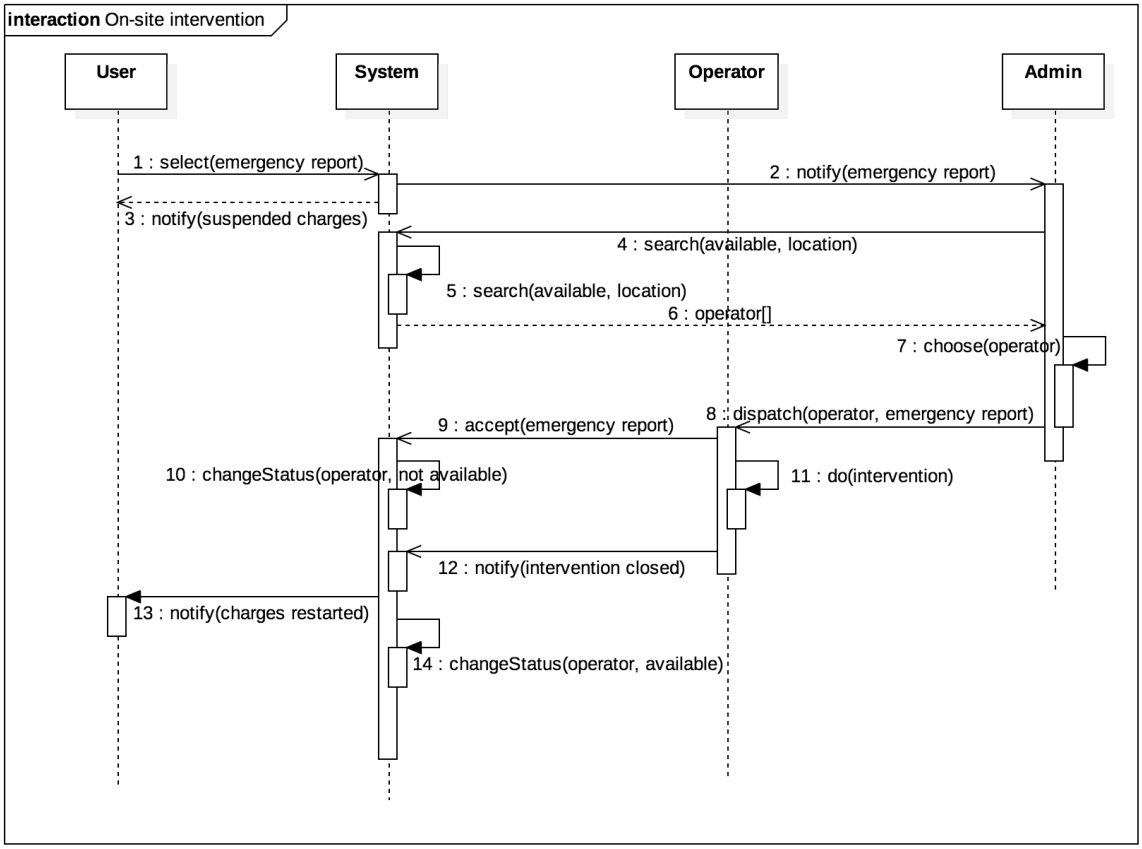
Special Requirements The new Operator cannot refuse the assignment if he is online and always accept it within 10 minutes. An Operator is always online when at work.

5.2 Class diagram

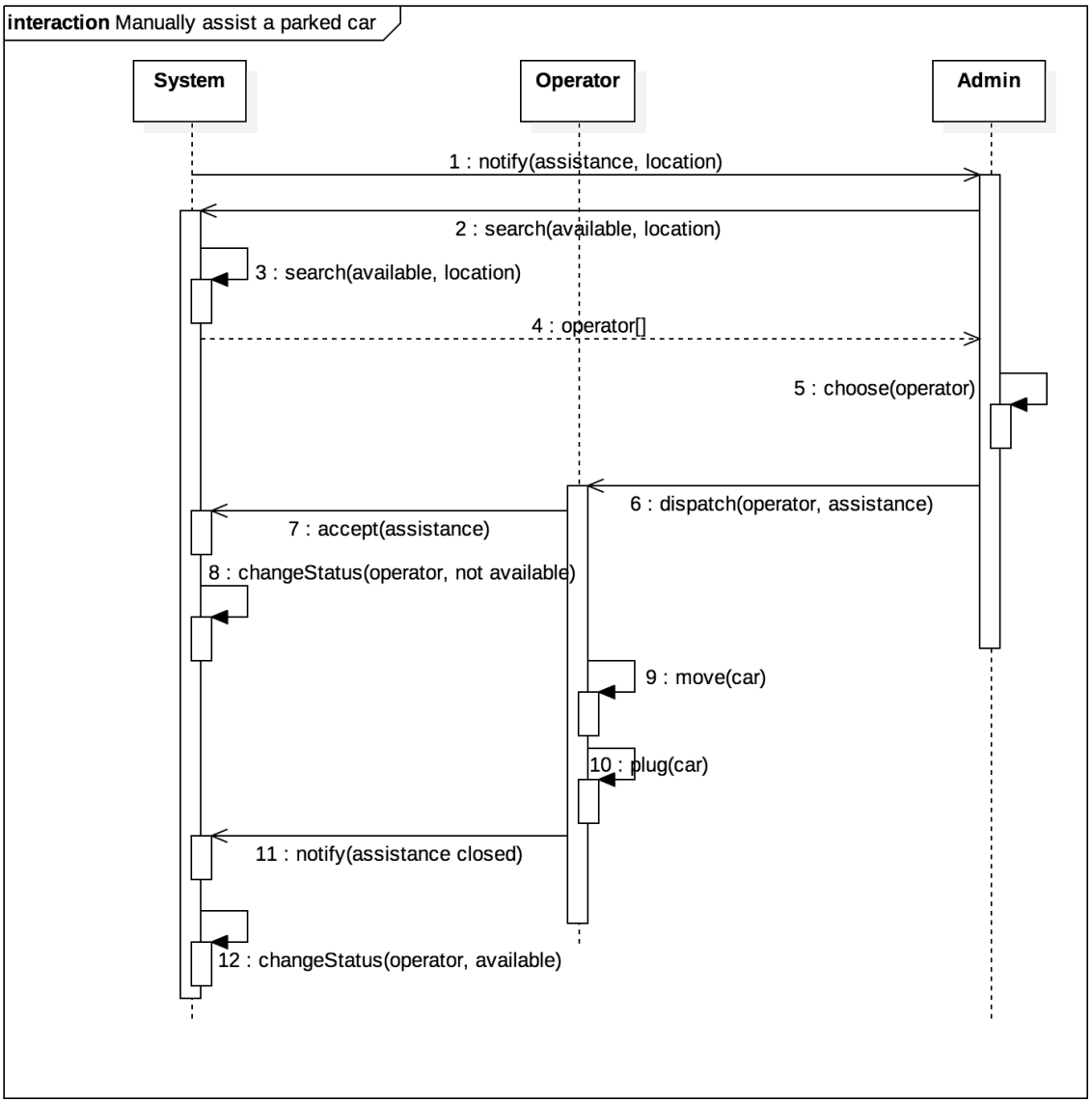


5.3 Sequence diagrams

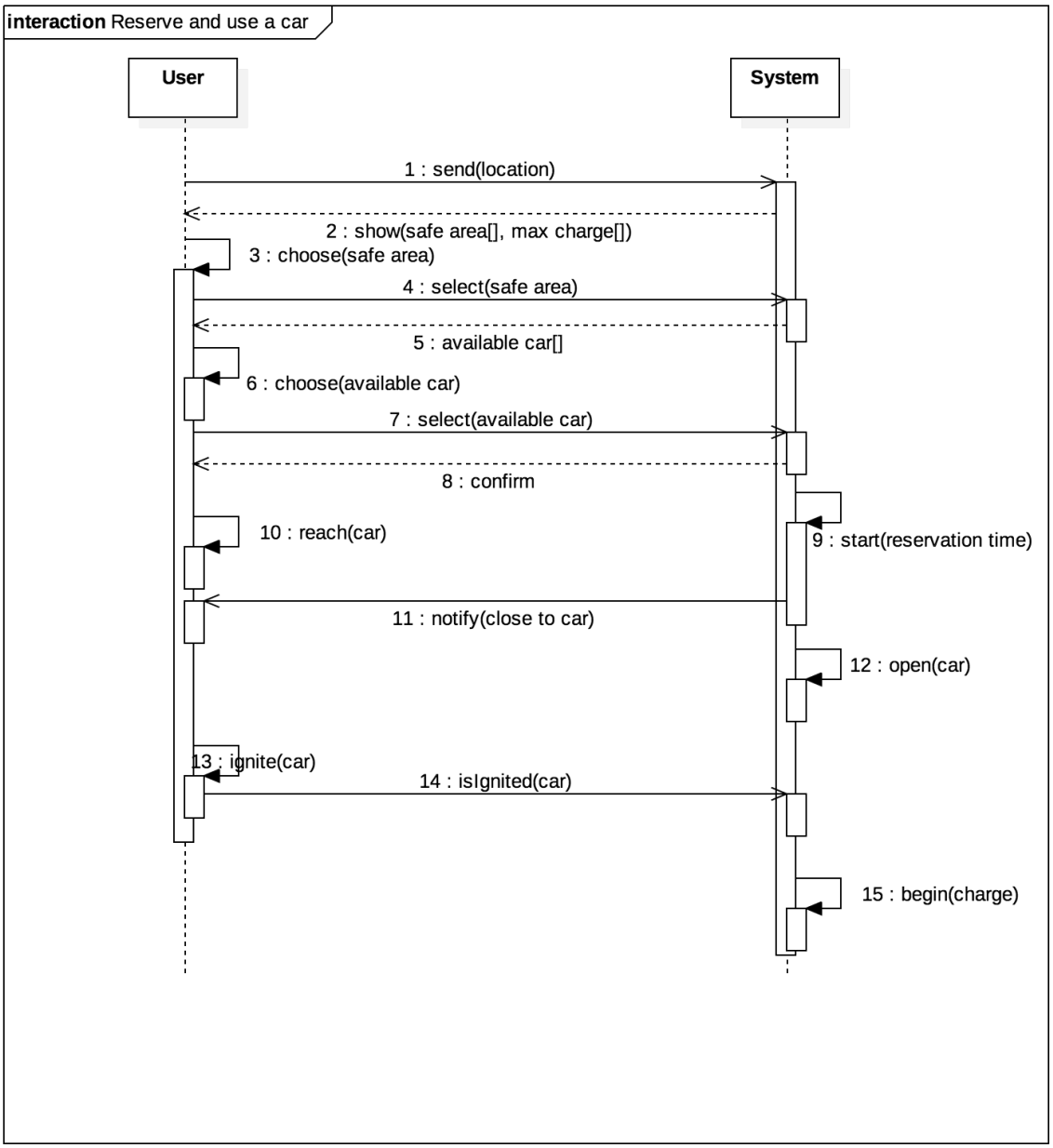
5.3.1 Sequence diagram 1: On-site intervention



5.3.2 Sequence diagram 2: Manually assist a parked car



5.3.3 Sequence diagram 3: Reserve and use a car



6 Alloy modeling

6.1 Model

The following model concerns the most characterizing features of the system. The model is intended as a static analysis, therefore it does not consider time or state transitions, but only invariant properties of the described world. We avoided to burden the model with trivial and non-significant details.

```
open util/boolean

/** Signatures **/

/* Atomic */
sig License{
isExpired: one Bool
}
sig Passenger {}

/* Internal actors */
sig Operator {}

/* Battery status */
enum BatteryStatus {
EmptyBattery, // < 3%
BatteryLow, // 3% - 20%
BatteryMedium, // 20% - 50%
BatteryHigh // > 50%
}

/* Emergency report enums */
enum ERStatus {
EROpen, ERDispatched, ERWip, ERClosed, ERCantClose
}
enum ERType {
ERAccident, EROnsite, ERNotOnsite
}

/* Car status */
enum CarStatus {
Available, Reserved, InUse, OutOfOrder
}

enum DistanceFromChargingArea {
Far, Near // Far: more than 3km; Near: otherwise
}
```

```
/* Discounts and Sanctions types */
abstract sig DiscountSanctionPerMinute { }
sig PassengersDiscount extends DiscountSanctionPerMinute { }
abstract sig DiscountSanctionWholeRide { }
sig HighBatteryDiscount extends DiscountSanctionWholeRide { }
sig PlugInDiscount extends DiscountSanctionWholeRide { }
sig MoneySavingOptionDiscount extends DiscountSanctionWholeRide { }
sig FarChargingAreaSanction extends DiscountSanctionWholeRide { }
sig LowBatterySanction extends DiscountSanctionWholeRide { }

/* User */
abstract sig GeneralUser {}
sig Guest extends GeneralUser {}
sig User extends GeneralUser {
license: one License,
banned: one Bool,
active: one Bool, // commodity - true if not banned and license not exipred
near: set Car
} {
active = False <=> (banned = True or license.isExpired = True)
}

/* Parking area */
abstract sig GeneralParkingArea {
capacity: one Int,
cars: set Car
} {
capacity > 0
#cars <= capacity
}
sig ParkingArea extends GeneralParkingArea {
distanceChargingArea: one DistanceFromChargingArea
}
sig ChargingArea extends GeneralParkingArea { }

/* Car */
sig Car {
battery: one BatteryStatus,
parkedIn: lone GeneralParkingArea,
isCharging: one Bool,
status: one CarStatus,
driverInside: one Bool,
locked: one Bool, // doors locked/unlocked
engineOn : one Bool
```

```
} {
isCharging = True => ( parkedIn != none and parkedIn in ChargingArea)
// isCharging only if in charging area
parkedIn = none <=> ( status = OutOfOrder or
( status = InUse and this[Ride<:car].timeWindowActive = False ) )
// car not parked can either be out of order or in use (exluded during time window)
status = Reserved => parkedIn != none // reserved only if parked
status in (Available + Reserved + OutOfOrder) => driverInside = False
// no driver inside if the car is not in use
engineOn = True => status in (InUse + OutOfOrder)
battery = EmptyBattery => engineOn = False
battery = EmptyBattery => status not in (Available + Reserved)
}

/* User interactions*/
sig Reservation {
user: one User,
car: one Car
}
sig Ride {
user: one User,
car: one Car,
moneySavingOption: one Bool,
moneySavingOptionSuggestion: lone GeneralParkingArea,
passengers: set Passenger,
timeWindowActive: one Bool,
chargesRunning: one Bool,
discSanctApplicableNow: set DiscountSanctionPerMinute,
discSanctApplicableRide: set DiscountSanctionWholeRide,
isStandard: one Bool
// ride finishes with the car being parked in a safe
// area and no emergency has occurred
} {
#passengers < 4 // capacity of cars (1 driver + 3 passengers)
moneySavingOptionSuggestion != none <=> moneySavingOption = True
isStandard = False <=> some er : EmergencyReport | er.user = user and er.car = car
}
sig EmergencyReport {
ride : one Ride,
user: one User,
assignedOp: lone Operator,
car: one Car,
status: one ERStatus,
type: one ERType
} {
```

```
user = ride.user
car = ride.car
assignedOp = none <=> status = EROpen // assignedOp empty iff status is EROpen
}

/** Facts **/

/* Structural constraints */

fact AttrbutePairings {
Car<:parkedIn = ~(GeneralParkingArea<:cars)
// Car::parkedIn = transpose of GeneralParkingArea::cars

no disjoint u1, u2 : User | u1.license = u2.license // license is personal
User.license = License // not consider licenses of people outside the system

all c : Car | ( c.status = Reserved <=> (some r : Reservation | c = r.car) )
all c : Car | ( c.status = InUse <=> (some r : Ride | c = r.car) )

all r : Ride | let u = r.user, c = r.car | ( c.driverInside  = True ) => ( u.near = c
}

fact exclusivity {
all c : Car | ( lone res : Reservation | res.car = c )
// every car is in 0..1 reservation
all c : Car | ( lone ride : Ride | ride.car = c )
// every car is in 0..1 ride
Reservation.car & Ride.car = none
// no car both reserved and in ride
all u : User | ( lone res : Reservation | res.user = u )
// every user has 0..1 reservation
all u : User | ( lone ride : Ride | ride.user = u )
// every user has 0..1 ride
Reservation.user & Ride.user = none
// no user with both a reservation and a current ride

all disjoint r1, r2 : Ride | r1.passengers & r2.passengers = none
// passengers no in more ride at the same time

all disjoint e1, e2 : EmergencyReport | e1.ride & e2.ride = none
// no rides with more than one emergency report

all disjoint e1, e2 : EmergencyReport | e1.assignedOp & e2.assignedOp = none
// operator can be assigned to one emergency report at a time
}
```

```
/* Requirements */

// G[1] The system allows guests to register; to complete the registration
// procedure the system sends a password to the guest as an access key.
fact RegistrationRequirements {
// none
}

// G[2] The system should enable a registered user to find the location of
// an available car within a certain distance from the user's location or
// from a specified address.
fact LocalizationRequirements {
// none
}

// G[3] The system enables user to reserve a single available car in a
// certain geographical region for one hour before the user picks it up.
// If the car is not picked up by that time, the reservation expires, the system
// tags this car as available again and it charges the user a fine of 1 EUR.
fact ReservationRequirements {
// R[3.5] reservation only for user with valid license
no userWithRes : Reservation.user | userWithRes.license.isExpired = True
// R[3.6] no banned user can reserve a car
no userWithRes : Reservation.user | userWithRes.banned = True
}

// G[4] The system should allow the user to employ a car in a proper and safe way.
fact RideRequirements {
// R[4.9] car unlocked only if user with reservation or which is
// using it is near it
all c : Car |
c.locked = False =>
( some u : User , r : Reservation | r.user = u and r.car = c and c in u.near ) or
( some u : User , r : Ride | r.user = u and r.car = c and c in u.near )
// R[4.10] car locked if user has exited it inside a safe area
all c : Car |
(
c.parkedIn != none and
c.driverInside = False and
( c in Ride.car => #( ((Ride<:car).c).passengers ) = 0 )
// c has no passengers (if it is in a ride, otherwise this is not considered)
=>
c.locked = True
)
```

```
// R[4.12] time window definition (partial coverage of the requirement)
all r : Ride | let c = r.car | r.timeWindowActive = True => c.parkedIn != none
// time window only if car in parking area
}

// G[5] The system charges the user for a predefined amount of money
// per minute. A screen on the car notifies the user of the current charges.
fact ChargesRequirements {
// none
}

// G[6] The system starts charging the user as soon as the car ignites.
// It stops charging them when the car is parked in a safe area and the
// user exits the car.
fact ChargingRequirements {
// general requirement
all r : Ride | let c = r.car | ( c.status = InUse and c.engineOn = True )
=> r.chargesRunning = True
// general requirement
all r : Ride | r.timeWindowActive = True => r.chargesRunning = False
// R[6.4] charges stops when user exits the car in a safe area
//(or haven't entered yet)
all r : Ride | let c = r.car | ( c.parkedIn != none and c.driverInside = False )
=> r.chargesRunning = False
}

// G[7] The system should encourage good user behaviour through the application
// of discounts to the fee per minute.
fact DiscountsRequirements {
// R[7.3] discount for rides with two or more passengers
all r : Ride | PassengersDiscount in r.discSanctApplicableNow
<=>
#r.passengers >= 2 and r.car.engineOn = True
// R[7.4] discount if more than 50% battery at end of a standard ride
all r : Ride | HighBatteryDiscount in r.discSanctApplicableRide
<=>
r.timeWindowActive = True and r.car.battery = BatteryHigh and r.isStandard = True
// R[7.5] discount if car plugged in at the end of the time window (approximation)
all r : Ride | PlugInDiscount in r.discSanctApplicableRide
<=>
r.timeWindowActive = True and r.car.isCharging = True
// NB: for simplicity here the ride is considered end when the time window
// is active. This does not affect the model.
}
```

```
// G[8] The system should discourage bad behaviour through the application
// of sanctions to the fee per minute.
fact SanctionsRequirements {
// R[8.1] sanction when car left at more than 3km from a charging area
all r : Ride |
FarChargingAreaSanction in r.discSanctApplicableRide
<=>
r.timeWindowActive = True and r.car.parkedIn in ParkingArea
and r.car.parkedIn.distanceChargingArea = Far
// R[8.2] sanction if car left with less than 20% of battery
all r : Ride |
 LowBatterySanction in r.discSanctApplicableRide
<=>
r.timeWindowActive = True and r.car.battery = BatteryLow
// NB: for simplicity here the ride is considered end when the time window
// is active. This does not affect the model.
}

// G[9] The system should provide an alternative usage mode for cars called
// money saving option. Besides aiding the user in saving money, this mode
// allows for a uniform distribution of cars throughout the city by suggesting the
// user where to park.
fact MoneySavingOptionRequirements {
// R[9.2] discount in money saving option if car returned in suggested area
all r : Ride |
MoneySavingOptionDiscount in r.discSanctApplicableRide
<=>
r.car.parkedIn = r.moneySavingOptionSuggestion
}

// G[10] The system allows the company to assist the users in case
// of need and take care of the cars.
fact AssistanceMaintenanceRequirements {
// none
}

/** Assertions **/

// Car assigned or reserved to the input User
fun carRelatedTo [u : User] : set Car {
(( Ride<:user ).u ).car + (( Reservation<:user ).u ).car
// (set of cars related through Ride or Res for u)
}

// 1 if the driver is inside the car; 0 otherwise
```

```
fun driverCountInCarCapacity [c : Car] : one Int {
c.driverInside = True =>1 else 0
}

assert carToSingleUser {
// no car assigned to more than one user
all disjoint u1, u2 : User |
( u1.carRelatedTo & u2.carRelatedTo ) != none
=>
u1.carRelatedTo != u2.carRelatedTo // if not both empty, then different)
}
check carToSingleUser

assert noReservationWithEmptyBatteryCar {
// no car reserved with empty battery
all r : Reservation | r.car.battery != EmptyBattery
}
check noReservationWithEmptyBatteryCar

assert driverNeverTrapped {
// driver can't be inside a car if the car is not assigned to him
no c : Car | c.status != InUse and c.driverInside = True
}
check driverNeverTrapped

assert carMaxCapacity {
// car can't have more than 4 person inside
all r : Ride | r.car.driverCountInCarCapacity + #r.passengers <= 4
}
check carMaxCapacity

/** Predicates **/

/* Domain assumptions */
pred DA {
// everything has already been represented with the model itself
}

pred show {
DA
}

run show
```

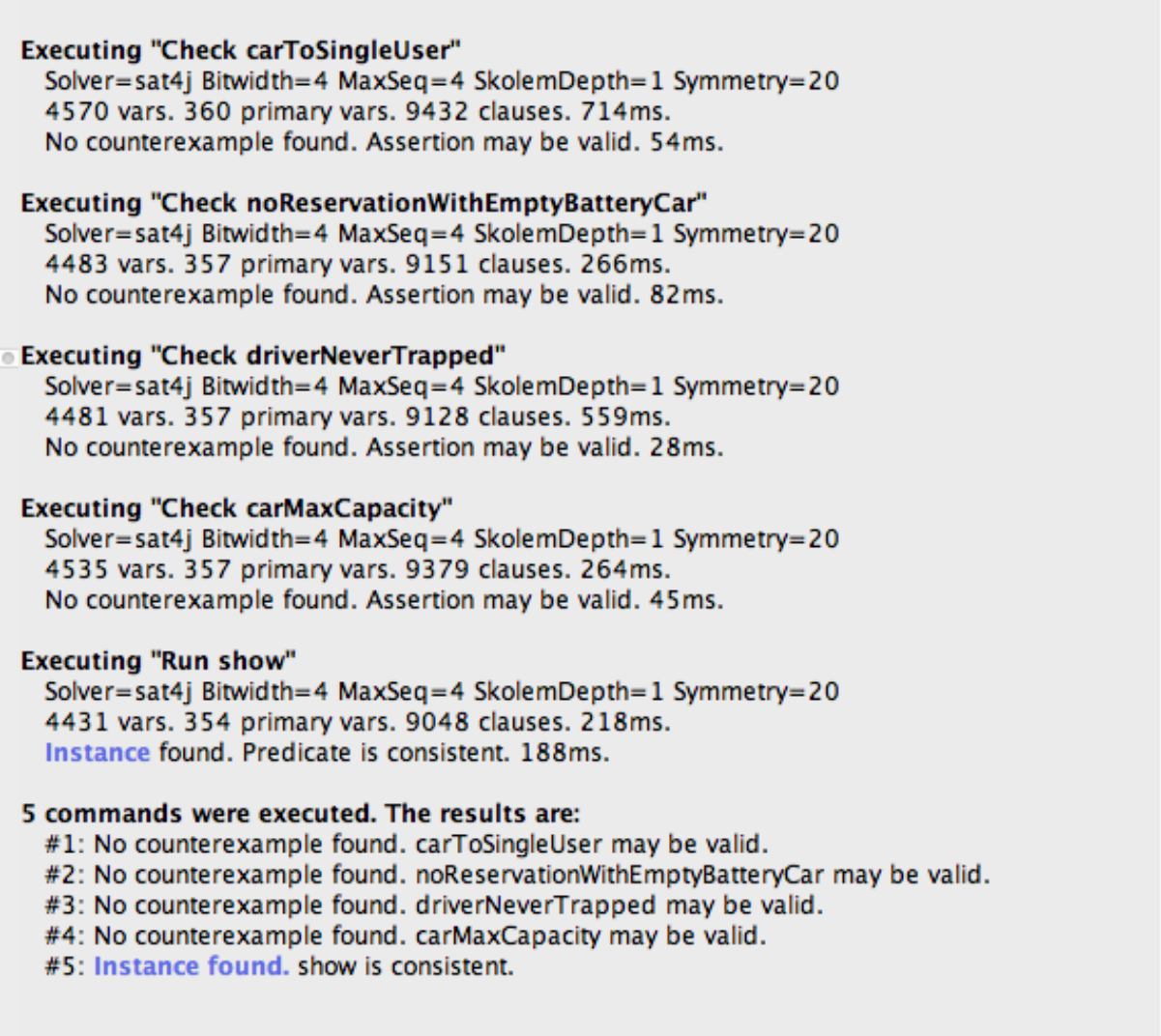
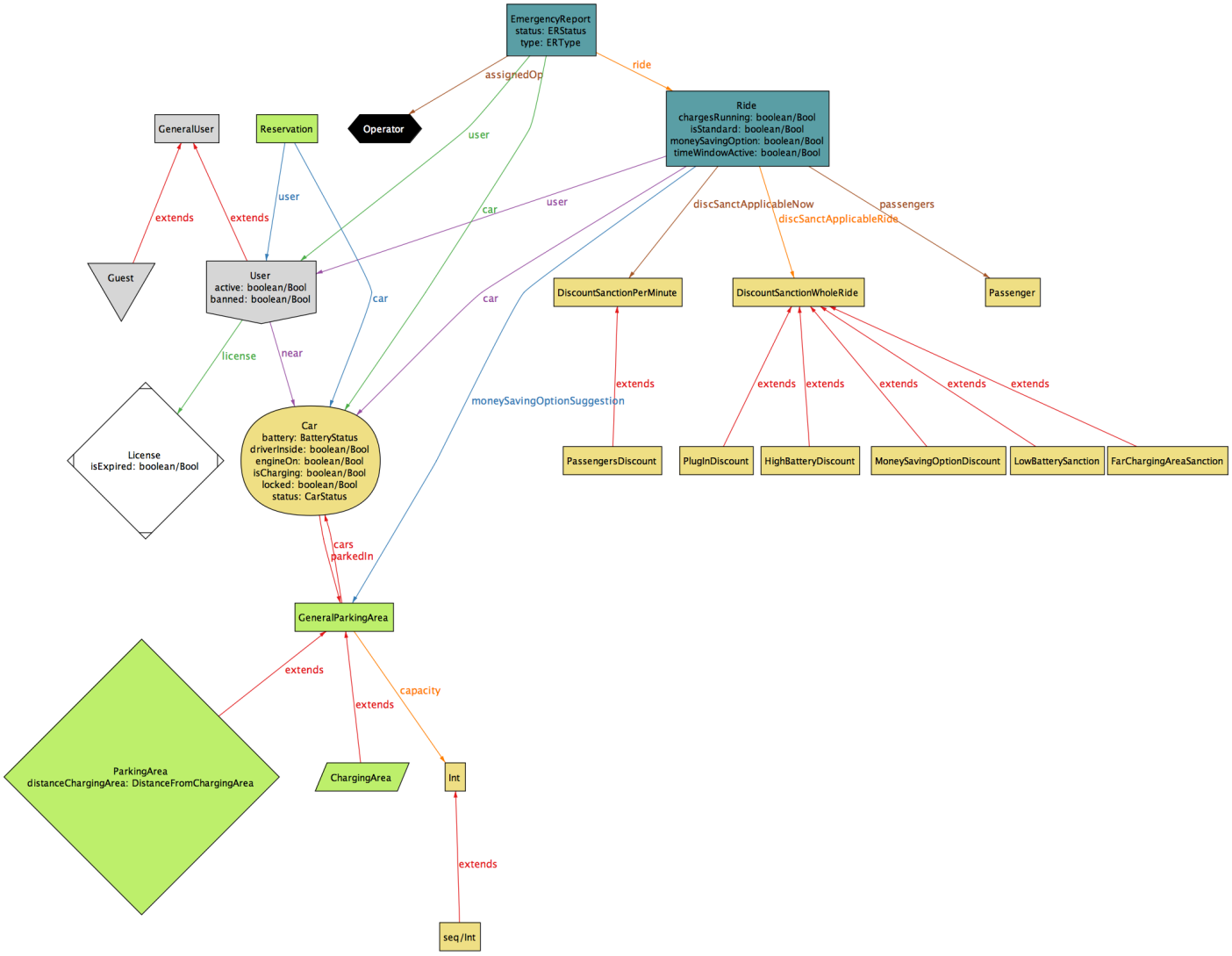


Figure 4: Result of the model analysis.

6.2 Alloy result

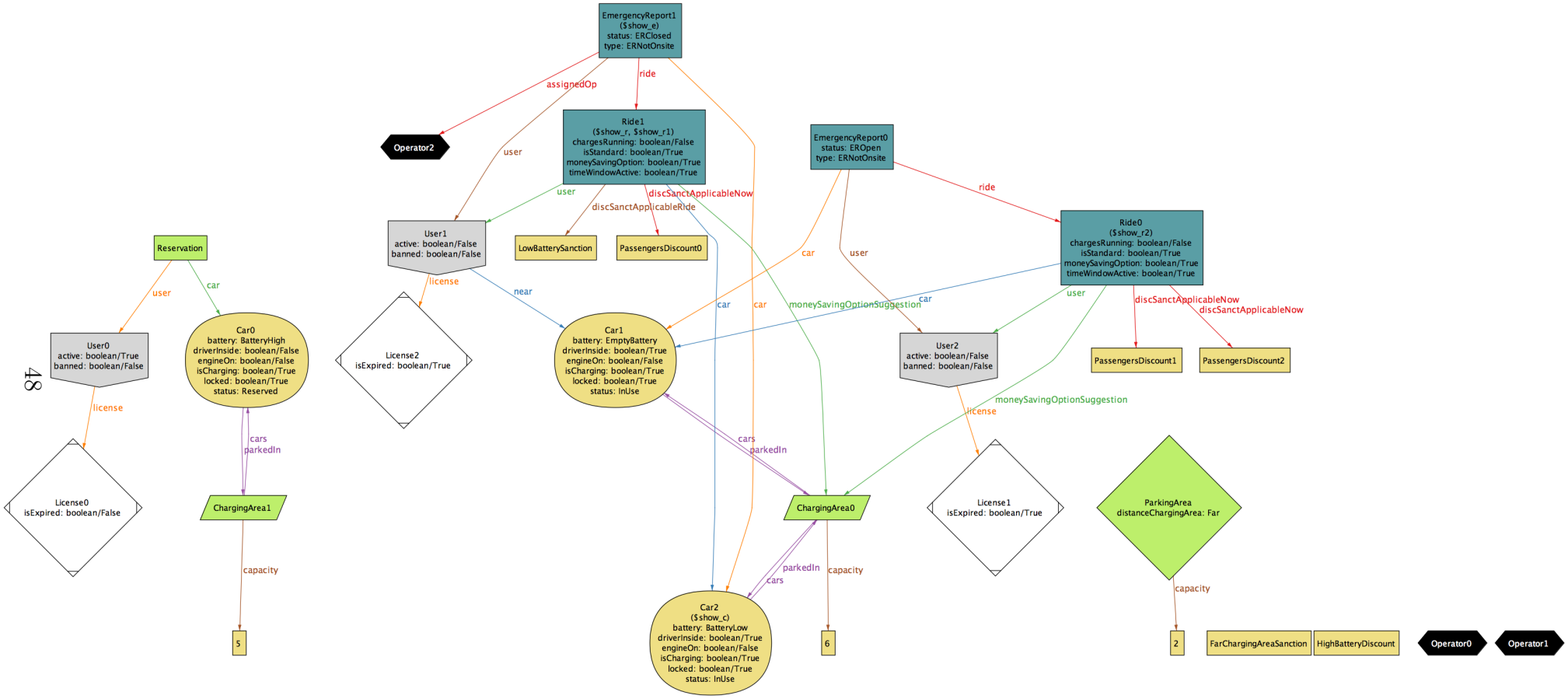
The model is consistent: see Figure 4.

6.3 Metamodel

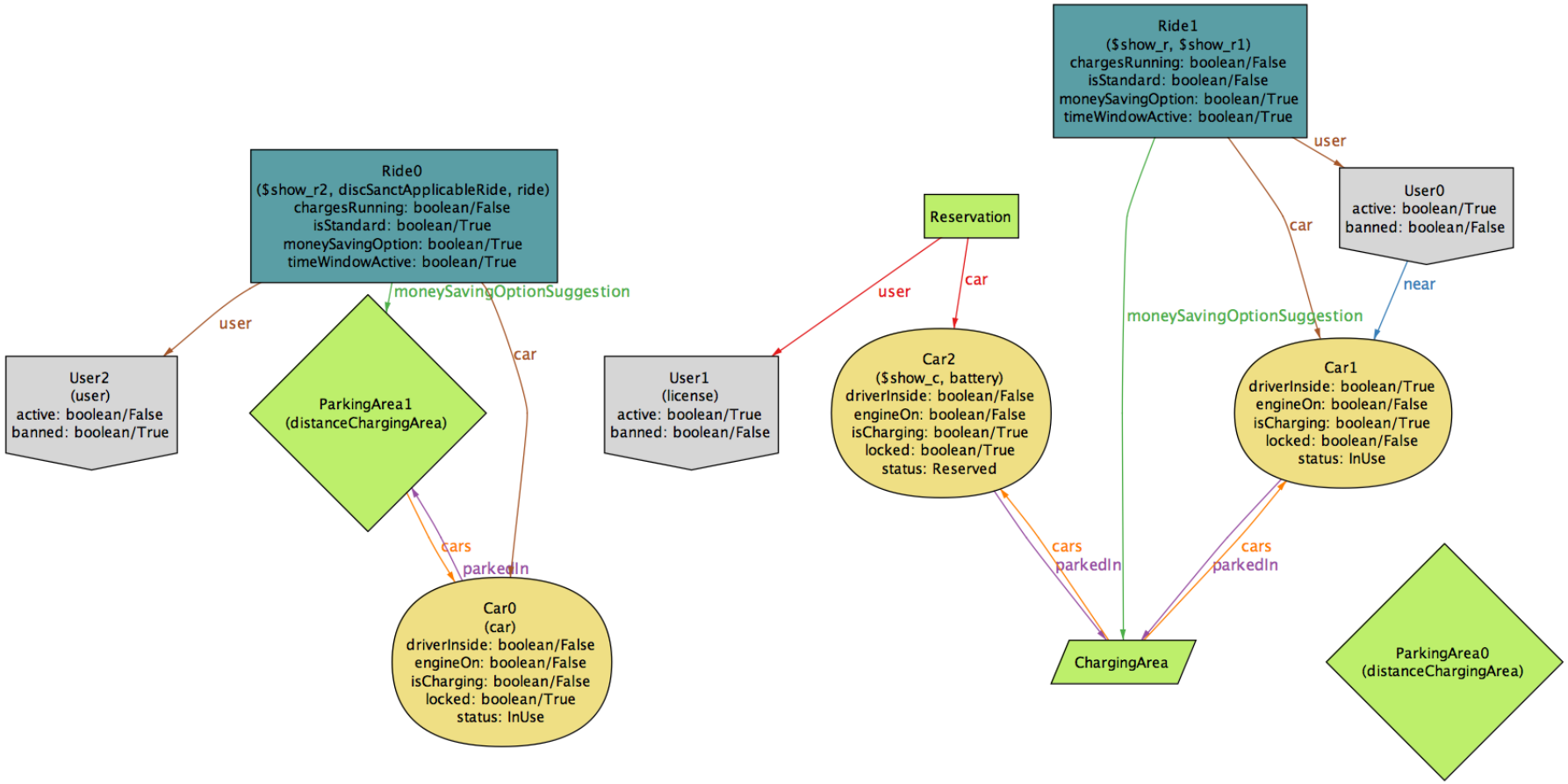


6.4 Worlds generated

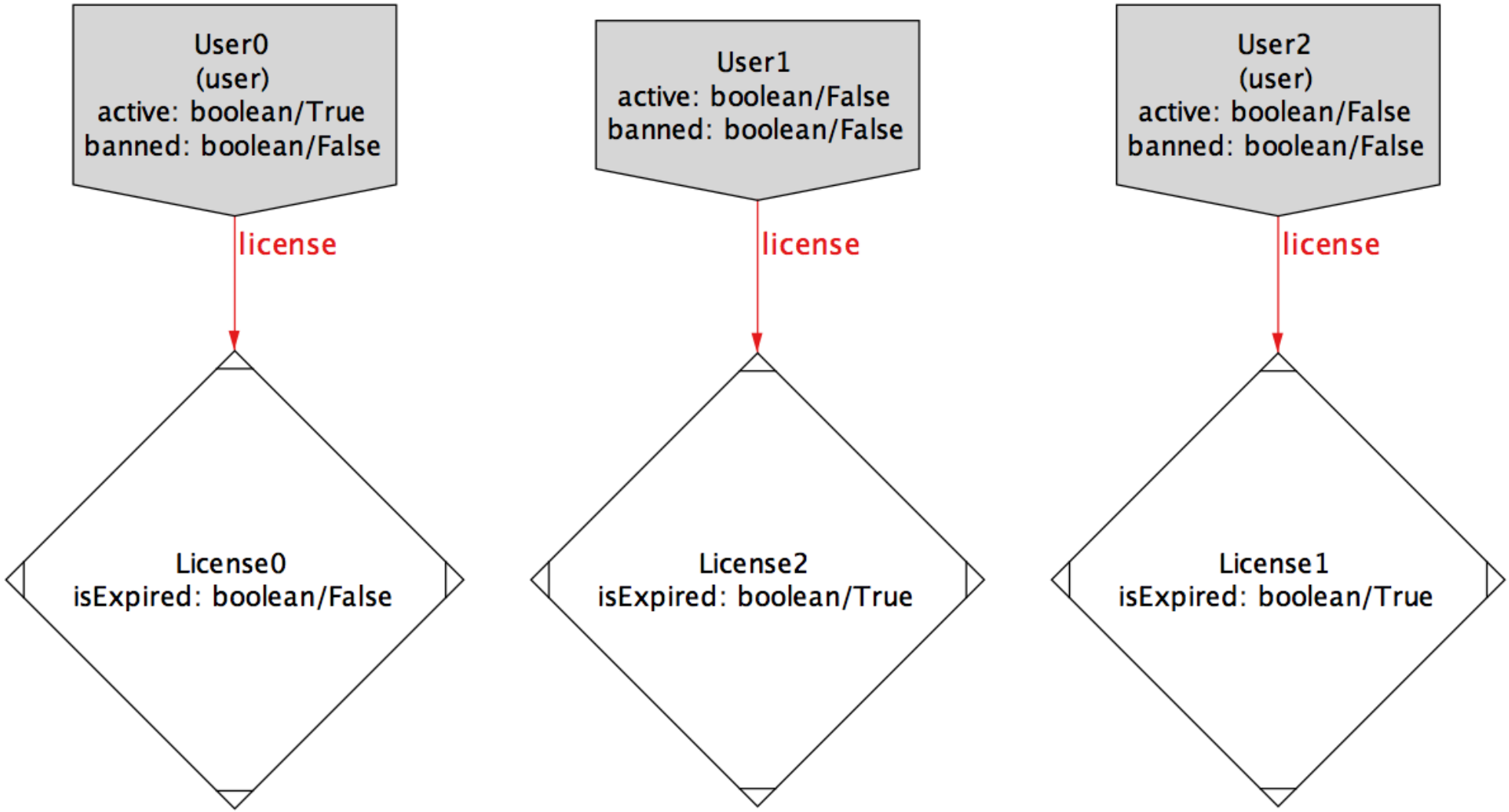
6.4.1 General world



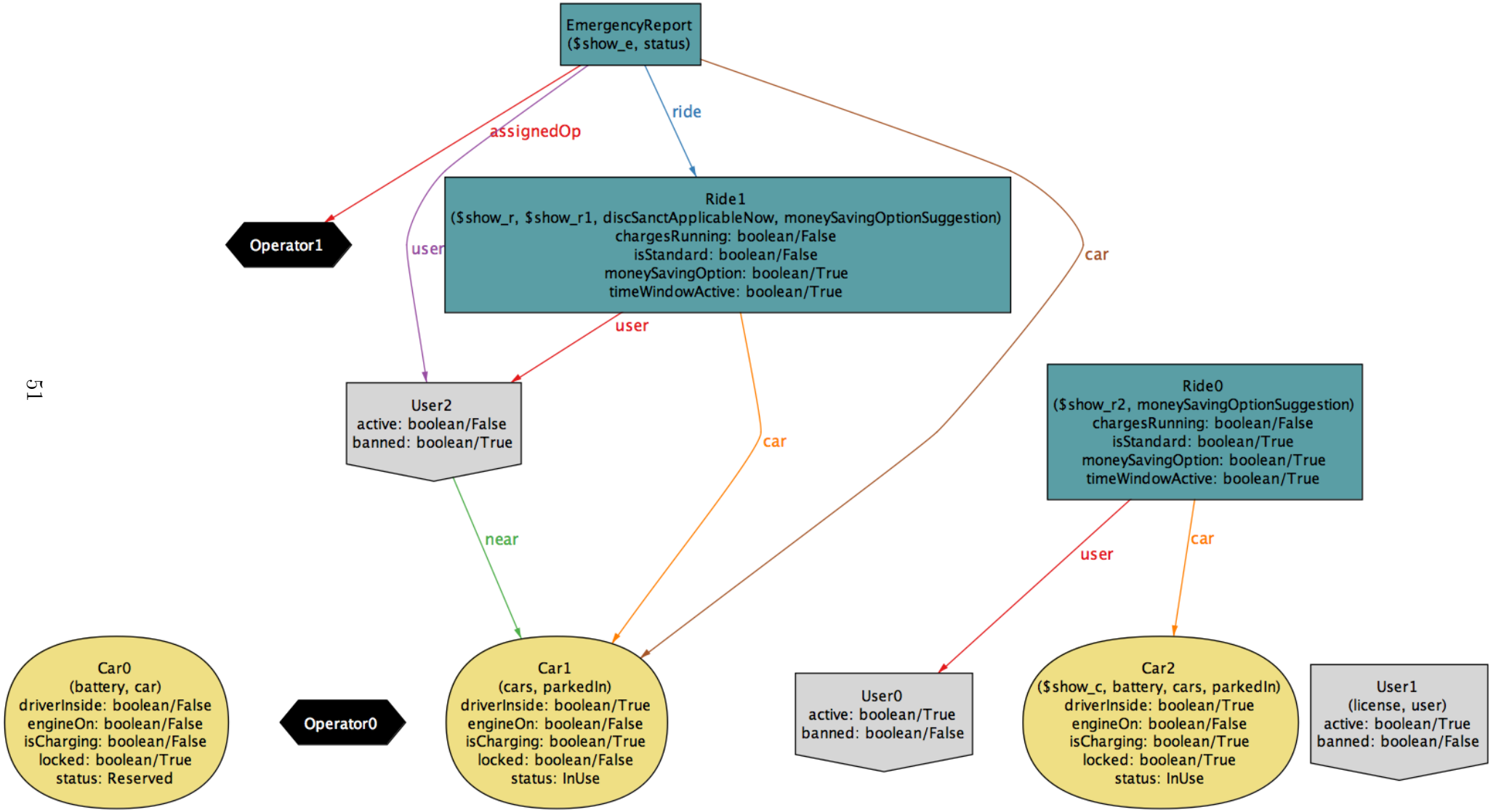
6.4.2 Ride-reservation projection



6.4.3 User licenses projection



6.4.4 Emergency report projection



7 Appendix

List of Figures

List of Tables

7.1 Used tools

For this assignment, we used the following tools:

Alloy We used the alloy tool to write the code and check the models for the specification.

LaTeX The group used LaTeX to structure the final document and to help with versioning.

Github We leaned on Github for versioning and coordinating synchronized work.

Toggl We used toggl to keep track of work hours.

Slack We used Slack for messaging and coordinating between us.

7.2 Hours of work

Abbud, Patricia around 25 hours of work;

Andreoli Andreoni, Maddalena around 54 hours of work;

Cudrano, Paolo around 68 hours of work.