Requirements Analysis and Specifications Document

POLITECNICO
MILANO 1863

Version 1.0

Luca Santini 808710
Riccardo Remigio 874939

Index

1.Introduction

1.1 Description of the given problem

We have to build a system for a car sharing service called PowerEnJoy, that permits a client to rent a car paying the service per minute.
The client can register to the system and get the access to it to use the services. After the access, the client can locate the available cars and can reserve one of them for a limited time. Once the user arrives to the car, that he reserved before, he can unlock and use it. When the client finishes to use the car, he parks it in a safe area and exit from it. The car will lock automatically and the system will stop charge the client.
The system also provides a set of discounts and fees to improve the behavior of the clients.
To manage the cars, there are many technicians that can plug the car into the power grids.

1.2 Scope

A person who have a driving license, and a valid payment method should be able to register himself to the system.
Therefore, a registered person should be able to get authenticated by the system. When the system authenticates a user, he should be able to localize all the available cars and to reserve one, if he isn't in the black list.
Now the user must reach the car before the reservation time expires, if it were to expire, the reservation will be cancelled and the user will be charged of the reservation fee; else the user should be able to unlock the car and take it.
Therefore, the user drives the car and the system charges him the correct amount of money established by PowerEnjoy's business rules, while the user is driving he should be able to check the current charging. When the user will want to stop the ride (and the charging) he will have to park the car into a safe area, get out of it and close the doors.
If the system will not be able to charge the user the money, he will be inserted the black list.

There are many discounts for the users that have a good behavior, these are established by the PowerEnJoy's business rules, we will specify it in the paragraph 1.3.1.
There will be many functionalities also for the technicians, to improve their organization. The technicians will be able to check the cars states and if another technician is already taking care of a car or if the car is plugged into a power grid.

1.3 Goals

G1. A person who has the right requirements[1] must be able to register himself to the system.
G2. A registered person must be able to authenticate himself to the system.
G3. A user must be able to localize the positions of the available vehicles.
G4. A user must be able to reserve an available vehicle, for a limited time[2].
G5. A user who has reserved a vehicle, must be able to use[3] it.
G6. The system must properly charge[4] for the user the cost of used services[5].
G7. The system must properly manage[6] the availability of vehicles.
G8. The system must simplify the organization[7] of the technicians in their work.

1.3.1 Notes

This notes integrate the definitions of the goals.

1- *Right requirements (for the registration):*
   To register to the system, you must provide your name, your surname, your date of birth, your birth place, your tax code. You must also provide a valid driving license, and a valid payment method.
2- *Limited time (for the reservation):*
   There is maximum 1 hour to reach the car until the reservation expires.
3- *Use a vehicle:*
   It means that the user can unlock the doors, enter in the car, and drive to his destination. The user now has to park in a safe area to end the ride.

4- *Properly charge the user:*
   To properly charge the user means that If he can pay, the system will charge him the correct amount of money depending on the discounts and the rates established by the business rules.
   If he can't pay, he will be insert in the black list and he can't use the service until he pays the last ride.
   The current business rules provide to charge the user for an amount of money per minute, and to improve good behavior there are some discounts.
   If the system detects the user took at least two other passengers onto the car, the system applies a discount of 10% on the last ride.
   If a car is left with no more than 50% of the battery empty, the system applies a discount of 20% on the last ride.
   If a car is left at special parking areas where they can be recharged and the user takes care of plugging the car into the power grid, the system applies a discount of 30% on the last ride.
   If a car is left at more than 3 KM from the nearest power grid station or with more than 80% of the battery empty, the system charges 30% more on the last ride to compensate for the cost required to re-charge the car on-site.
5- *Services (that charges for the user):*
   The services are the reservation and the rental of a car.
   The user has to pay a fee for the reservation only if the when he reserves a car the reservation time expires. In this case the user has to pay 1€ with the current rules.
6- *Properly manage the availability of vehicles:*
   If a vehicle is reserved or is in use it is not available. Also a vehicle with battery level lower than a certain minimum threshold is not available. In all the other cases the vehicle is available.
   Then the system has to manage the vehicles states to guarantee these conditions.
7- *Simplify the organization of the technicians:*
   It means that the technicians have to know what are the vehicles with low battery not in charge, and if another technician is already taking care of it.

1.4 Domain assumption

These are the conditions we suppose to hold in our world.

- GPS always indicates the right position.
- The GPS of all the vehicles is always working.
- The vehicles usable by the clients are always working, especially during the whole ride.
- The battery level of a vehicle is always enough high to complete the ride of the user.
- The user who reserved the vehicle is the same who will use it.
- When a free technician notices a low battery vehicle, he takes care of it.
- A technician that has plugged a vehicle into a power grid changes the vehicle state from “In processing” to “available”
- A user parks always in a safe area

1.5 Glossary

These are some words that are used frequently in our descriptions, we explain the precise meaning to improve the expressive clearness.

- System: It’s the whole system that we have to plan to get working the system.

- Registration: It’s a system procedure that associates and stores people’s data to the system, and then provides to the user a password that has to be used to authenticate himself.

- Authentication: It's a system procedure that allows to recognize a registered user by their credentials.

- Vehicles/Cars: The electric cars employed by PowerEnJoy for car-sharing.

- State of vehicles: (Available, Reserved, Not available, Low battery, In processing)
    - Available: means that the vehicle is available for reservation and ready for the use, with the battery level more than a minimum threshold.
    - Reserved: means that the vehicle is reserved by a user, that will reach the vehicle.
    - In use: means that a user is actually using this vehicle.
    - Not available: means that the vehicle is not available for the users, it is a generic state that can be useful in particular situations.
    - Low battery: means that the vehicle is not available for the users because the battery level is lower than the minimum threshold, so a technician will take care of plugging it into the nearest power grid.
    - In processing: means that a technician is taking care of make it available as soon as possible.

- Reserve a vehicle: A user can reserve a vehicle for at most one hours, after that, the reservation expires and the user have to pay a fee of 1€.

- Use a vehicle: A user that has reserved a car must be able to unlock the door, and to drive the car until he reaches his destination. When the user wants to end the ride, he has to park the car and get out of it.

- Technicians: They are PowerEnJoy employees, that take care of the operation of all the vehicles.

- User: A person who is registered to the system, and who is authenticated.

- Registered person: A person who is registered to the system but it isn't still authenticated.

- Safe area: it is an area in which a user can park to end the ride.

2. Actors identifying

- Guest: it is a person who wants to join to the car-sharing services of PowerEnJoy, he can only register to the system, after that he can logging into the system
- User: It is a person authenticated, he can localize, reserve a car and use it.
- Technician: They are PowerEnJoy employees, they can check the state of the vehicles and they have to fix the problems by plugging the cars into the power grids.

3. Requirements

Supposed that the domain assumption made in the paragraph 1.3, from the goals, written in the paragraph 1.2, we can derive these requirements.

Below each goal we write the relative requirements

G1. A person who has the right requirements must be able to register himself to the system:

- The system must be able to verify the completeness and correctness of the data provided by the person who wants to register
- The system must provide a sign up functionality that gives a password to the user to access the system
- The system must allow the sign up only if the data provided by the person are correct.

G2. A registered person must be able to authenticate himself to the system:

- The system must be able to check the correctness of the user's login credentials
- The system permits the access to the user only if the login credentials are correct

G3. A user must be able to localize the position of the available vehicles:

- The system must know the position of the vehicles
- The system must know the state of the vehicles
- The system must be able to receive the position communicated by the user
- The system must provide a functionality that shows the available vehicles to the user

G4. A user must be able to reserve an available vehicle:

- The system must provide a functionality that permits to the user to reserve a vehicle
- Only the vehicles shown to the user can be reserved
- The system must know if the user has a vehicle's reservation
- The system doesn't allow to reserve a car if the user is in the black list
- The system permits to reserve a vehicle only if the user hasn't already reserved another vehicle

G5. A user who has reserved a vehicle, must be able to use it:

- The system must provide a functionality to unlock or lock the vehicle's doors

G6. The system must properly charge customers the cost of used services:

- The system must know the number of passengers
- The system must know the level of charge of the vehicles
- The system must know if the car is ignited or not
- The system must know if the vehicle is in charge or not
- The system must provide a functionality that shows to the user, during the ride, the current charge

- The system must provide a functionality that calculates the amount that the user has to pay according to the discounts and rates established
- The system must be able to charging the amount calculated to the user
- The system inserts the user in the black list if it is not able to pay

G7. The system must properly manage the availability of vehicles:

- The system must know the state of the vehicles
- The system must know the level of charge of the vehicles
- The system must know if the car is ignited or not
- The system must know if the vehicle is in charge or not
- The system must be able to change the state of the vehicles
- The system must provide a functionality that manage the availability of the vehicles in the correct way

G8. The system must simplify the organization of the technicians in their work

- The system must know the state of the vehicles
- The system must know if the vehicle is in charge or not
- The system must provide a functionality that shows to the technicians all the vehicles with their state and if the car is in charge or not.
- The system must be able to change the state of the vehicles based on what the technicians want

3.1 Functional Requirements

Now that we have defined all the requirements, we will resume the functionality available to actors

Guest can:

- Sign up
- Log in

User can:

- Locate an available car
- Check the battery level of the available vehicles
- Reserve an available car
- Unlocks the door of a Reserved car
- Check the current charges (obviously during the ride)

Technician can:

- Locate all the cars
- Check the state of all the cars
- Check if a car is plugged into a power grid
- Check if another technician is taking care of a car

4. Scenario identifying

Here are described many scenarios to explain how the system works, with practical examples.

Scenario 1

Luca wants to register to PowerEnJoy's services. He is provided of a driving license that permits him to drive the electric cars of the car-sharing service. He is also provided by a payment method that is accepted by PowerEnJoy. So he opens the registration page of PowerEnJoy's website and fills the registration form with his personal information. Once he has finished he receives a password that he has to use to log in the system.
Now Luca can log in the system and can use all the services that PowerEnJoy reserves to its users.

Scenario 2

After many hours of lecture, Marco wants to come back home from the university. Marco is lucky, because his university and his home are both covered by the safe area of the PowerEnJoy's car-sharing. So Marco logs into the system and searches a car near his position. Now he reserves that car and reaches the position on foot before the reservation's time expires, then he unlocks the car and drives the car.
During the path Marco can check his current charges. When he arrives at home he parks the car and get out of it, to end the ride and to stop the charging.

Scenario 3

John is in a conference and he is very tired, so he wants to go home quickly as soon as the conference over. The place where the conference is taken and John's home are both located in the safe area of PowerEnJoy service, so when the conference is almost over John reserves a car. But when the conference is over John has to reach the position of the car, that isn't very close to him. So before John can reach the car the reservation's time expires and unfortunately John have to pay the fee and he has to looking for another car.

Scenario 4

Alice is a technician of PowerEnJoy and this morning has to go to work. When she arrives in the office, she can check the state of all the vehicles, so she finds many "low battery" cars not in charge. She chooses one and changes its state in "In processing", then she takes the key of this car from the office and reaches the position of the car. She unlocks it and drives the car to a free power grid and plugs in the car. Now the car is in charge, Alice changes the state of the car in "Available" and the system automatically set the state in "low battery" until the battery level is upper than the minimum threshold. Alice can now return to the office to take care of another car.

Scenario 5

Matt and his two friends are in the city center and they want to go to the cinema. Matt is a user of PowerEnJoy and he notice that there is a car near their, so he decides to reserve it. They take that car and go to the cinema, when they arrive at cinema they find a power grid where they can plug in their car. Therefore, they park and plug in the car and they should have a discount of 10% because the system has detected two passengers and a 30% of discount for have plugged the car into the power grid, but the system grants them only the bigger one (30% of discount on the whole ride)

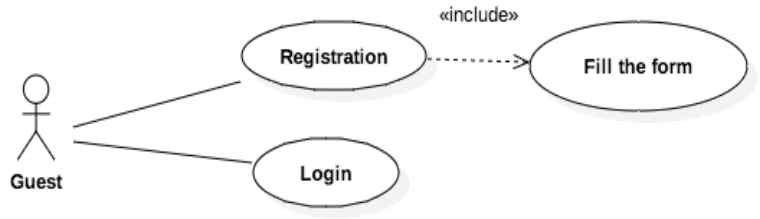
5. UML Models

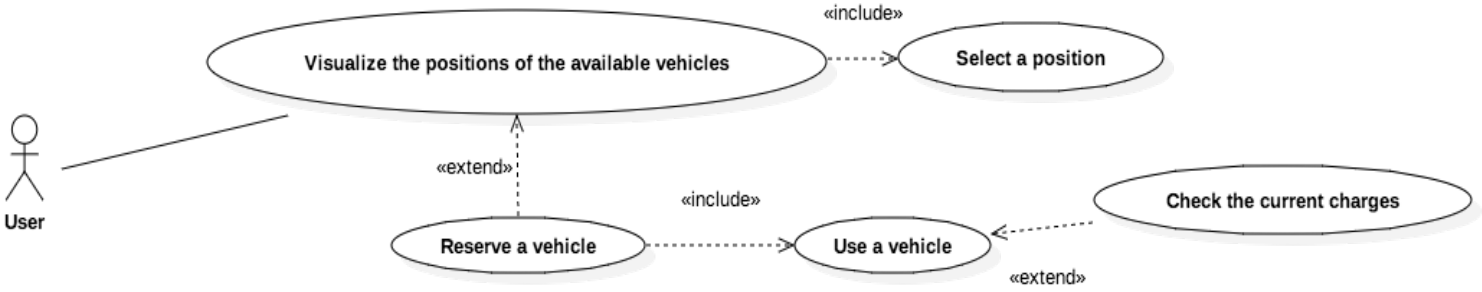
5.1 Use case diagram

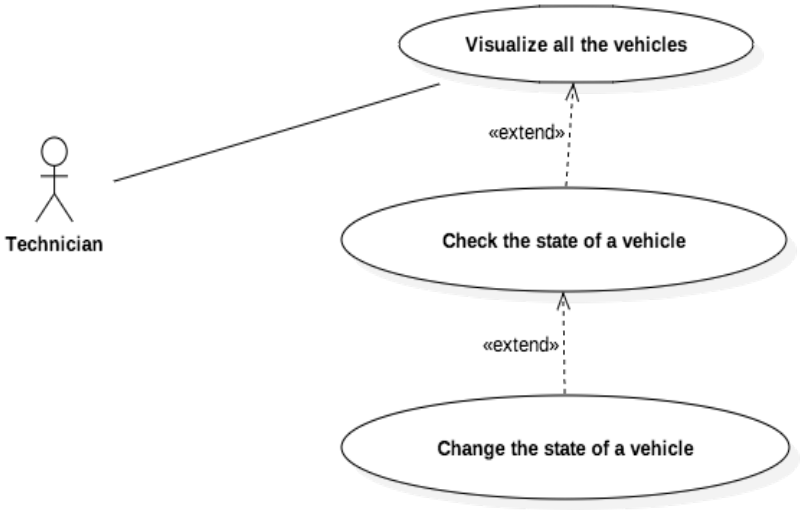
We can derive some uses cases from the precedent descriptions of the system

- Registration
- Login
- Visualize the positions of the available vehicles
- Reserve a vehicle
- Check the current charge
- Visualize all the vehicle (for the technician)
- Check the state of a vehicle
- Change the state of a vehicle

Here is the use case diagram







5.2 Use case description

We describe the use case of “Registration”

NAME	**Registration**
ACTORS	Guest
ENTRY CONDITIONS	The guest isn’t registered to the website
FLOW OF EVENTS	The guest: 1. connects to the system 2. selects the registration procedure
EXIT CONDITIONS	The guest accesses to the compilation form
EXCEPTIONS	An exception can be caused if a guest tries to register by using an username that is already used, or if he doesn’t have a valid payment method or a valid driving license

We describe the use case of “Login”

NAME	Login
ACTORS	Guest
ENTRY CONDITIONS	Guest has succesfully registered to the system
FLOW OF EVENTS	The guest: 1. connects to the system 2. inserts his username and the password, received after the registration, in a form 3. sends the form
EXIT CONDITIONS	The guest receives a notification for the success of the operation and enters the system as user
EXCEPTIONS	If the username or the password are incorrect the system notifies the error to the guest

We describe the use case of “Visualize the available vehicles”

NAME	Visualize available vehicles
ACTORS	User
ENTRY CONDITIONS	User is logged in successfully
FLOW OF EVENTS	1. The user selects a position 2. The system shows to the user all the available car near his position
EXIT CONDITIONS	The user can select an available vehicle shown by the system or exit the system
EXCEPTIONS	If there aren’t vehicle available at the moment

We describe the use case of “Reserve available vehicle”

NAME	Reserve available vehicle
ACTORS	User
ENTRY CONDITIONS	The user must have selected an available vehicle
FLOW OF EVENTS	1. The system shows to the user the position and the level of charge of the selected vehicle
EXIT CONDITIONS	Either the user confirms the reservation , the system changes the state of the selected vehicle to unavailable and notifies the user for the success of the operation or the user can exit the system
EXCEPTIONS	If the user is in the black list, the system notifies him that he can’t reserve a vehicle

We describe the use case of “Fill the form”

NAME	Fill the form
ACTORS	Guest
ENTRY CONDITIONS	The guest must have started a registration procedure
FLOW OF EVENTS	The guest: 1. inserts the personal data: name, surname, birth date, birth place and tax code 2. inserts his driving license 3. inserts the payment informations 4. sends the form to the system
EXIT CONDITIONS	The guest receives a confirmation and a password to access the system
EXCEPTIONS	If the guest doesn’t compile all the fields of the form or some fields are incorrect, he will receive a notification for the errors and will not receive the password

We describe the use case of “Use reserved vehicle”

NAME	Use reserved vehicle
ACTORS	User
ENTRY CONDITIONS	The user must have reserved a vehicle
FLOW OF EVENTS	The user: 1. arrives at the reserved vehicle 2. unlocks the vehicle through the system 3. ignites and uses the vehicle 4. parks the vehicle 5. turns off and exits the vehicle
EXIT CONDITIONS	The system calculates the cost of the ride, charges the user and changes the state of the used vehicle to available
EXCEPTIONS	When the system charges the user, if he hasn’t enough money, it puts him in the blacklist

We describe the use case of “Visualize all the vehicle”

NAME	Visualize all the vehicles
ACTORS	Technician
ENTRY CONDITIONS	Technician is authenticated to the system as a technician
FLOW OF EVENTS	1. The technician connects to the system 2. The system shows to the technician all the car
EXIT CONDITIONS	The technician can select a vehicle shown by the system or exit the system
EXCEPTIONS	No exceptions

We describe the use case of “Check the state of a vehicle”

NAME	Check the state of a vehicle
ACTORS	Technician
ENTRY CONDITIONS	The technician must have selected a vehicle
FLOW OF EVENTS	1. The system shows to the technician the position, the state, the level of charge of the selected vehicle
EXIT CONDITIONS	The technician can decide to change the state of the car or exit the system
EXCEPTIONS	No exceptions

We describe the use case of “Change the state of a vehicle”

NAME	Change the state of a vehicle
ACTORS	Technician
ENTRY CONDITIONS	The technician must have checked the state of a vehicle
FLOW OF EVENTS	1. The system shows to the technician some options to change the state of the selected vehicle 2. The technician chooses the state in which the vehicle will change
EXIT CONDITIONS	The system changes the state of the vehicle in that chosen by the technician and confirms the success of the operation
EXCEPTIONS	No exceptions

We describe the use case of “Check the current charges”

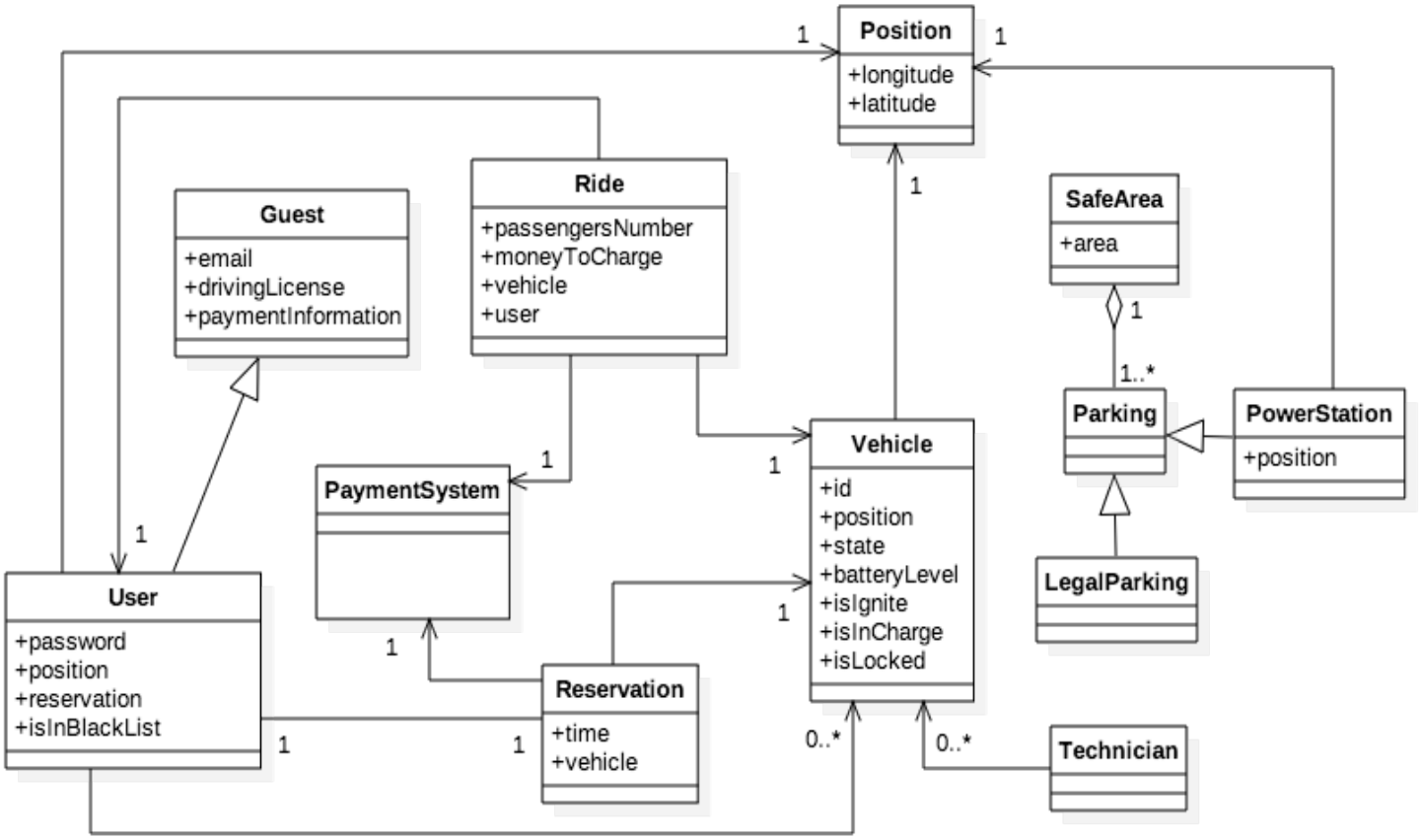
NAME	Check the current charges
ACTORS	User
ENTRY CONDITIONS	The user is using a vehicle
FLOW OF EVENTS	1. The user can look at the screen of the vehicle to retrieve information about the amount of the charging
EXIT CONDITIONS	There isn’t an exit condition, because the user can see the current charges by looking the screen in the vehicle
EXCEPTIONS	No exceptions

We describe the use case of “Select a position”

NAME	Select a position
ACTORS	User
ENTRY CONDITIONS	The user is visualizing the available vehicles
FLOW OF EVENTS	1. The user inserts his position through an address or a GPS system
EXIT CONDITIONS	If the user inserts the address, he has to confirm the choice, otherwise the system takes the position automatically
EXCEPTIONS	No exceptions

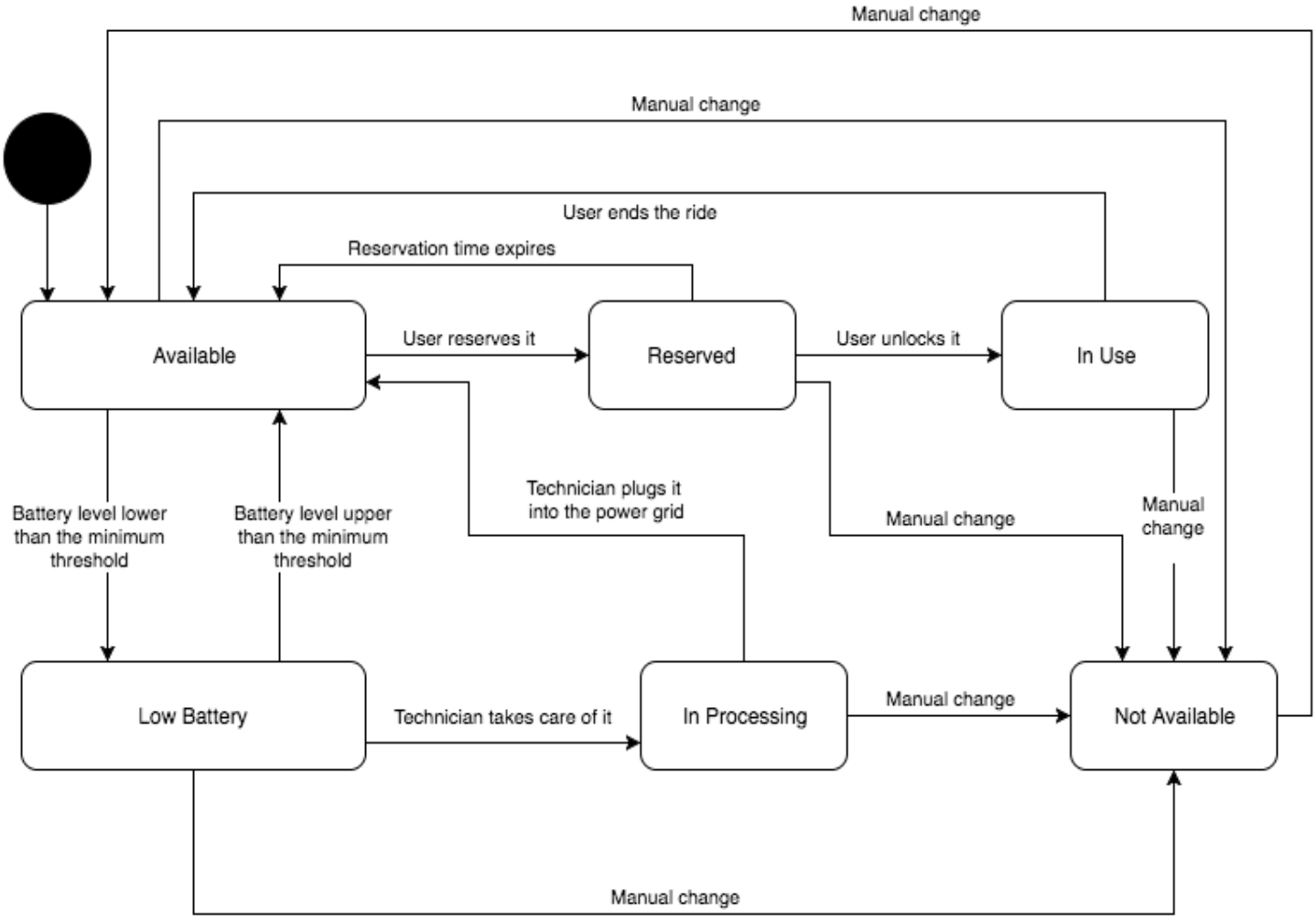
5.3 Class diagram

We constructed a class diagram to model our system and all its components.
In the next chapter we will verify the consistency of this class diagram with Alloy Analyzer.



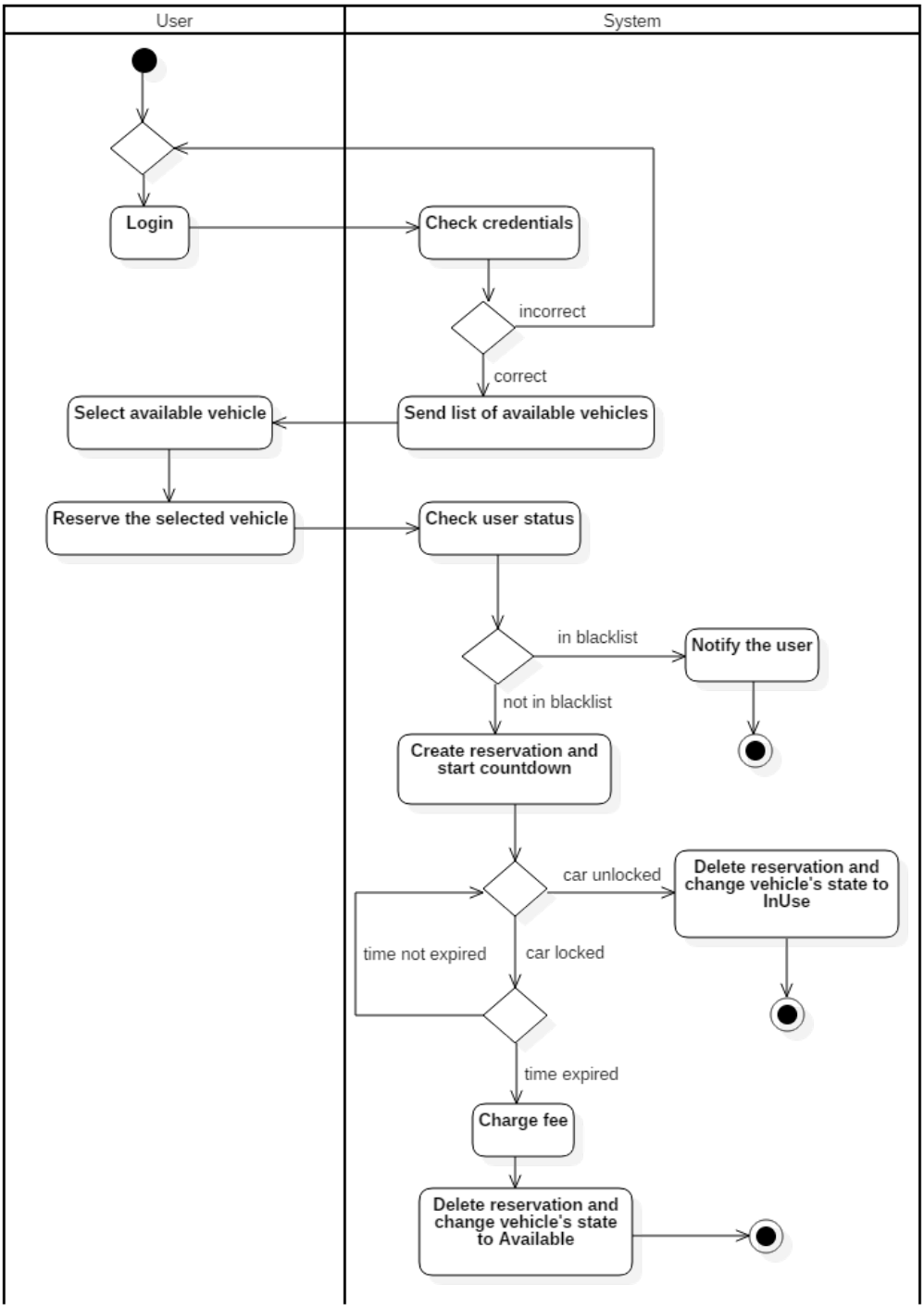
5.4 State chart diagram

We drew a state diagram that explains which are the states of the vehicles and how these may change.



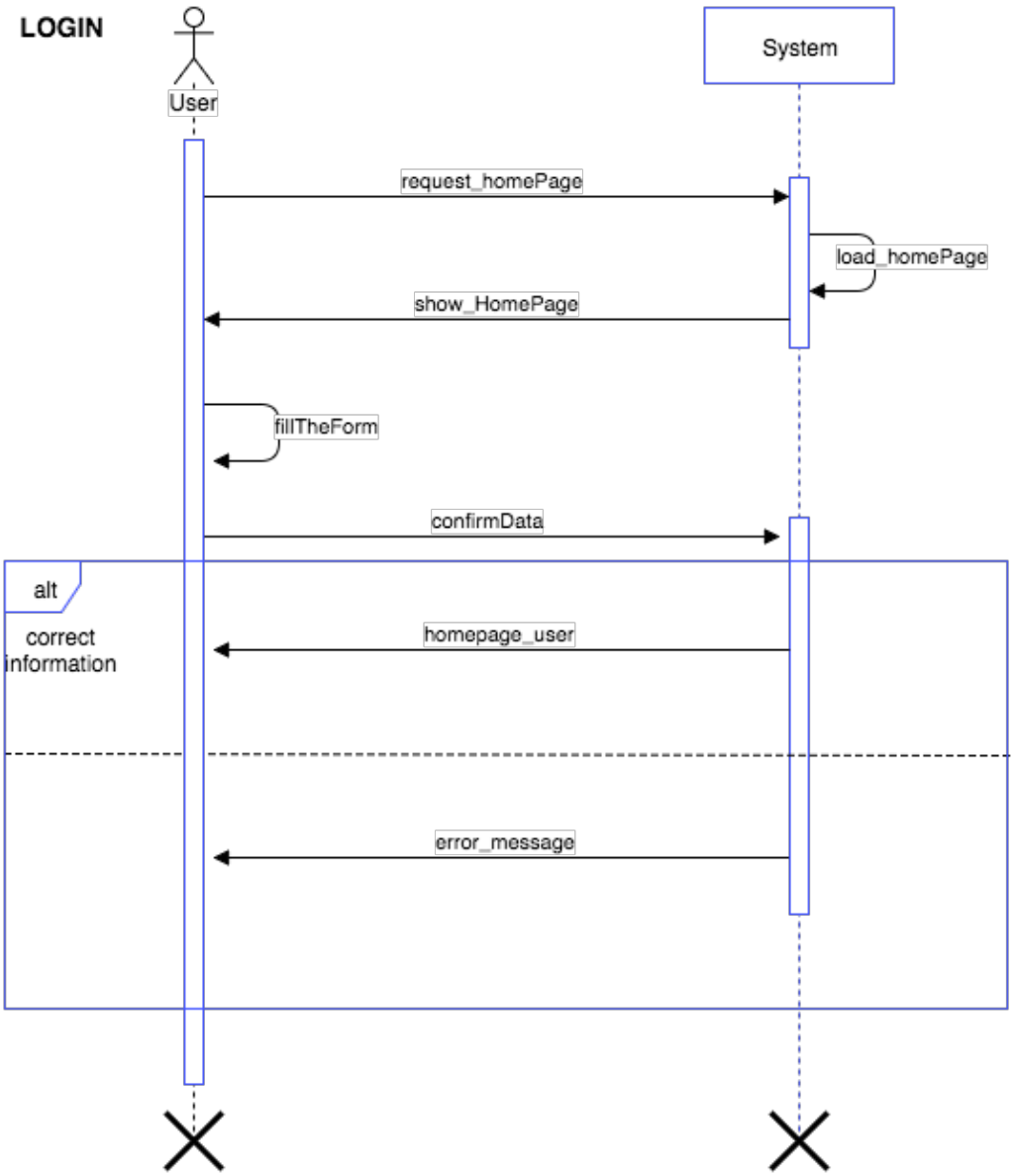
5.5 Activity Diagram

We drew an activity diagram to explain the reservation procedure.



5.6 Sequence Diagram

We drew a sequence diagram to explain the interaction between the user and the system during the login procedure



6. Alloy Modelling

We tested through the alloy analyzer the consistency of our model, here is the code:

```
open util/boolean


// SIGNATURES
sig Position{}

one sig SafeArea{
    position: set Position
}

// a vehicle can be parked in a PowerStation also if the vehicle is not inCharge
sig PowerStation{
    position : one Position,
    vehicles : set Vehicle,
    capacity : one Int
}{
    capacity >= #vehicles
    capacity < 5 // For istance
    vehicles.position = position
}

// We have ignored many useless information for the user like email, driving license
sig User{
    position: one Position,
    reservation: lone Reservation,
    isInBlackList: one Bool
}{
    isInBlackList.isTrue implies #reservation=0
}

sig Technician {
    isTakingCareOf: lone Vehicle
}{
    isTakingCareOf.state in InProcessing
}
```

```
sig Vehicle{
    position: one Position,
    state: one VehicleState,
    batteryLevel: one Int,
    isIgnited: one Bool,
    isInCharge: one Bool,
    isLocked: one Bool
}
{
    batteryLevel>=0 && batteryLevel<=100
}


sig Reservation{
    time: one Int,
    vehicle: one Vehicle
}{
vehicle.state in Reserved
time>0
time<60
}

sig Ride{
    passengersNumber: one Int,
    vehicle: one Vehicle,
    user: one User
}
{
    passengersNumber>0 && passengersNumber<=5
    vehicle.state in InUse
    #user.reservation=0
    user.isInBlackList.isFalse
}


abstract sig VehicleState{}
one sig Available extends VehicleState{}
one sig Reserved extends VehicleState{}
one sig InUse extends VehicleState{}
one sig NotAvailable extends VehicleState{}
one sig LowBattery extends VehicleState{}
one sig InProcessing extends VehicleState{}

//FACTS

fact reservationHasOneUser{
    all disj u1,u2: User | u1.reservation!=u2.reservation
    all r:Reservation | one u:User | u.reservation=r
}

fact vehicleReservedHasOneReservation{
    all disj r1,r2: Reservation | r1.vehicle!=r2.vehicle
    all v:Vehicle | v.state in Reserved => one r:Reservation | r.vehicle=v
}


fact userHaveOneRide{
    all disj r1,r2: Ride | r1.user != r2.user
}

fact vehicleInUseHasOneRide{
    all v:Vehicle | v.state in InUse => one r:Ride | r.vehicle=v
}


fact availableVehicle {
    all v1: Vehicle | v1.state in Available implies (v1.batteryLevel>20 and v1.isIgnited.isFalse and
    v1.isLocked.isTrue)
}
```

```
fact reservedVehicle {
    all v1: Vehicle | v1.state in Reserved implies ( v1.batteryLevel>20 and v1.isIgnited.isFalse)
}

fact inUseVehicle {
    all v1: Vehicle | v1.state in InUse implies (v1.isIgnited.isTrue and v1.isInCharge.isFalse)
}

fact lowBatteryVehicle {
    all v1: Vehicle | v1.state in LowBattery implies (v1.batteryLevel<20 and
    v1.isIgnited.isFalse and v1.isLocked.isTrue)
}

fact inProcessingVehicle {
    all v1: Vehicle | v1.state in InProcessing implies (v1.batteryLevel<20 and
    v1.isInCharge.isFalse)
}

fact aCarIsAlwaysParkedInASafeArea{
    all v1:Vehicle | v1.isIgnited.isFalse implies v1.position in SafeArea.position
}

fact aPowerStationIsAlwaysInASafeArea{
    all p:PowerStation | p.position in SafeArea.position
}

fact aVehicleInChargeIsInAPowerStation {
    all v:Vehicle | v.isInCharge.isTrue implies one p:PowerStation | v in p.vehicles
}

fact ifAUserIsInAVehicleHeHasTheSamePosition{
    all r:Ride | r.vehicle.position = r.user.position
}


fact thePositionOfAPowerStationIsUnique {
    all p1,p2 :PowerStation | p1!=p2 implies p1.position != p2.position
}

fact technicianWork{
    all v:Vehicle | v.state in InProcessing implies one t:Technician | v in t.isTakingCareOf
}

// ASSERTIONS

assert allTheVehiclesNotIgnitedAreInASafeArea{
    all v:Vehicle | v.isIgnited.isFalse implies v.position in SafeArea.position
}


assert numberOfTheRidesAreCorrect{
    #Ride = #{v:Vehicle | v.state in InUse}
}


assert numberOfTheReservationsAreCorrect{
    #Reservation = #{v:Vehicle | v.state in Reserved}
}


assert blackListUser{
    all r:Ride | r.user.isInBlackList.isFalse
    all u:User | u.isInBlackList.isTrue implies u.reservation = none
}

assert numberOfTechniciansIsGreaterOrEqualThanInProcessingVehicles{
    #Technician >= #{v:Vehicle | v.state in InProcessing}
}
```

```
check allTheVehiclesNotIgnitedAreInASafeArea for 8 Int
check numberOfTheRidesAreCorrect for 8 Int
check numberOfTheReservationsAreCorrect for 8 Int
check blackListUser for 8 Int
check numberOfTechniciansIsGreaterOrEqualThanInProcessingVehicles for 8 Int

pred ShowVehiclesPossibleStates{
#Vehicle = 6
#User = 3
#Technician = 2
some v:Vehicle | v.state in Available
some v:Vehicle | v.state in Reserved
some v:Vehicle | v.state in NotAvailable
some v:Vehicle | v.state in InUse
some v:Vehicle | v.state in InProcessing
some v:Vehicle | v.state in LowBattery
}

// the 3 vehicles are all parked in the powerStation but can be not plugged in
pred ShowAFullPowerStation{
    #PowerStation=1
    #Vehicle = 3
    one p:PowerStation | p.capacity = #p.vehicles and p.capacity=3
}

pred Show{}


run ShowVehiclesPossibleStates for 6 but 8 Int
run ShowAFullPowerStation for 3 but 8 Int
run Show for 4 but 8 Int
```

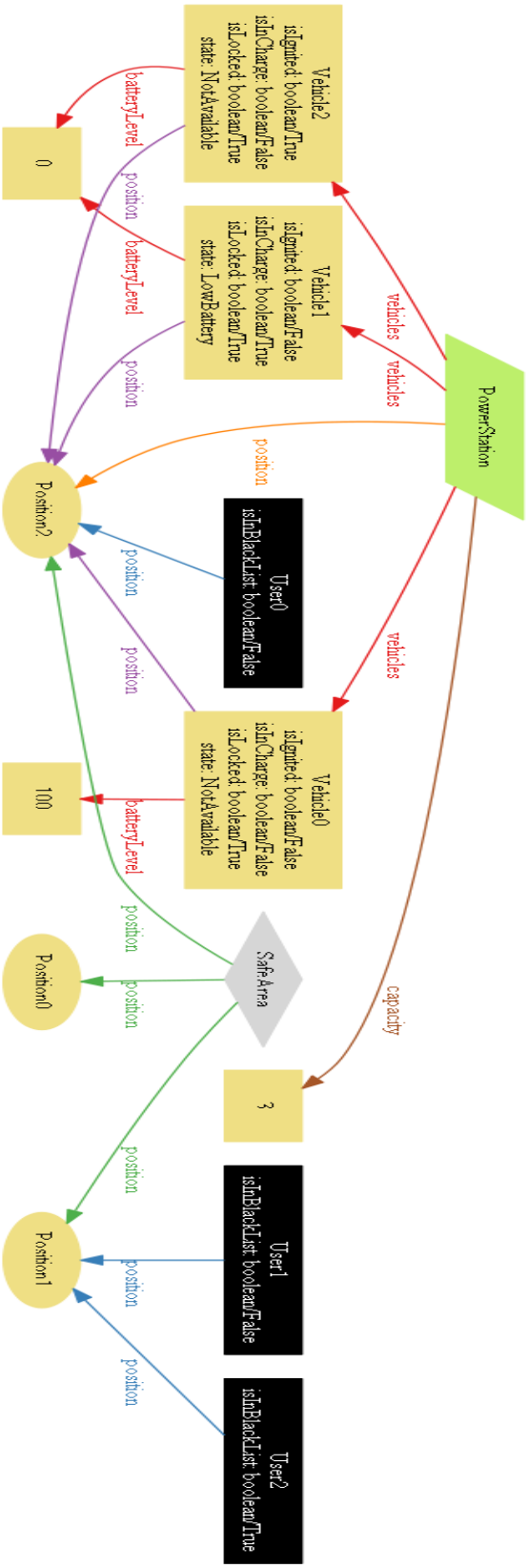
```
8 commands were executed. The results are:
  #1: No counterexample found. allTheVehiclesNotIgnitedAreInASafeArea may be valid.
  #2: No counterexample found. numberOfTheRidesAreCorrect may be valid.
  #3: No counterexample found. numberOfTheReservationsAreCorrect may be valid.
  #4: No counterexample found. blackListUser may be valid.
  #5: No counterexample found. numberOfTechniciansIsGreaterOrEqualThanInProcessingVehicles may be valid.
  #6: Instance found. ShowVehiclesPossibleStates is consistent.
  #7: Instance found. ShowAFullPowerStation is consistent.
  #8: Instance found. Show is consistent.
```

6.1 World Generated

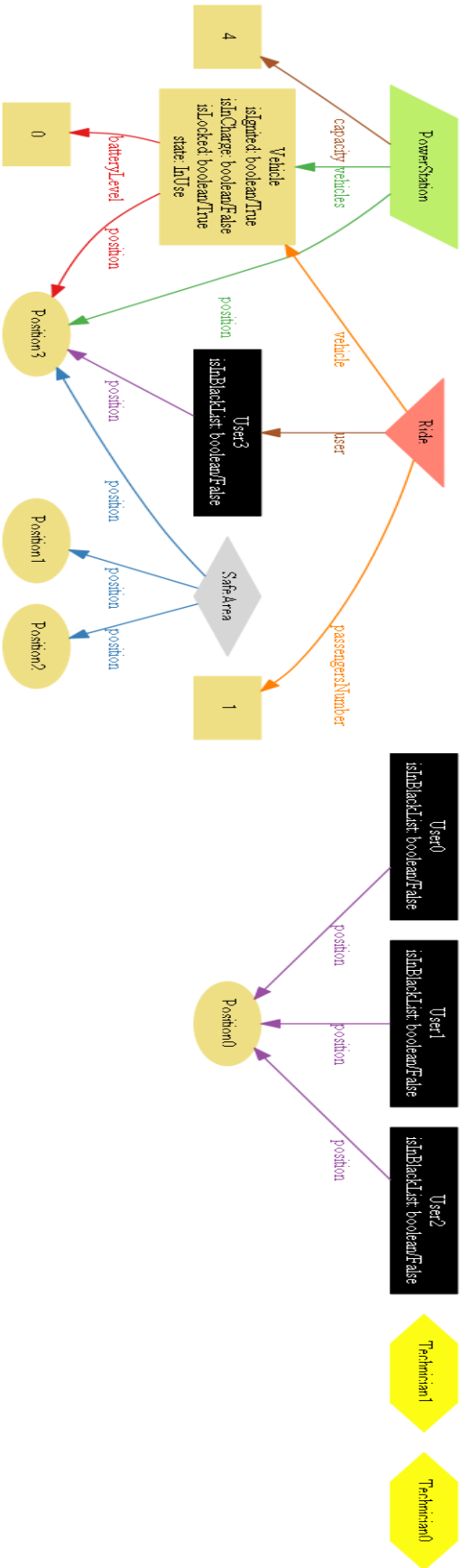
We created this world to test all the states of vehicles by forcing the generation of only six vehicles, one for each state, and by forcing the generation of three users. One for the "Reserved" vehicle, one for the vehicle "inUse" and the last one that must be free, because there aren't enough vehicles to use or reserve. We also forcing the generation of two technicians, one for the "InProcessing" vehicle and one that must be free.

Vehicle0
isIgnited: boolean/False
isInCharge: boolean/False
isLocked: boolean/True
state: LowBattery
batteryLevel
8
Vehicle1
isIgnited: boolean/True
isInCharge: boolean/False
isLocked: boolean/True
state: InProcessing
batteryLevel
1
Technician1
isTakingCareOf
Vehicle2
isIgnited: boolean/True
isInCharge: boolean/False
isLocked: boolean/True
state: InUse
batteryLevel
15
position
position
position
position
vehicle
user
passengersNumber
Ride
User2
isInBlackList: boolean/False
2
position
Position
position
capacity
PowerStation
SafeArea
position
position
position
position
User0
isInBlackList: boolean/False
Technician0
reservation
vehicles
Vehicle5
isIgnited: boolean/False
isInCharge: boolean/False
isLocked: boolean/True
state: Available
batteryLevel
32
position
User1
isInBlackList: boolean/True
vehicles
position
Vehicle3
isIgnited: boolean/False
isInCharge: boolean/False
isLocked: boolean/True
state: NotAvailable
batteryLevel
96
position
Reservation
56
time
vehicle
Vehicle4
isIgnited: boolean/False
isInCharge: boolean/False
isLocked: boolean/True
state: Reserved
batteryLevel
21

We create this world to show a possible condition of a power station in which there are three vehicles, one of them in charge, that fill the power station capacity. It is possible, like in the example, that there are some vehicles that are parked in the power station without being in charging



This is a generic world created by the show predicate of alloy, to prove the whole consistency of the model.



7.Used Tools

The tools we used to created this RASD are:

- Star UML
- Github
- Draw.io
- Alloy Analyzer
- GraphViz
- Microsoft Word

Hours of work

These are approximatively the time we spent to make all the analysis and to write this document

Luca Santini

- 02/11/16: 4h
- 04/11/16: 4h
- 06/11/16: 2,5h
- 07/11/16: 5h
- 08/11/16: 4h
- 10/11/16: 5h
- 11/11/16: 4h
- 12/11/16: 8h
- 13/11/16: 11h

Riccardo Remigio

- 02/11/16: 4h
- 04/11/16: 4h
- 05/11/16: 2h
- 06/11/16: 3h
- 08/11/16: 4h
- 10/11/16: 3h
- 11/11/16: 4h
- 12/11/16: 8h
- 13/11/16: 11h