



POLITECNICO DI MILANO
MSC COMPUTER SCIENCE AND ENGINEERING

SOFTWARE ENGINEERING 2
ACADEMIC YEAR 2016-2017

Requirements Analysis and Specification Document

PowerEnJoy

Authors:

Melloni Giulio 876279

Renzi Marco 878269

Testa Filippo 875456

Reference Professor:

MOTTOLA Luca

Release Date: November 13th, 2016

Version 1.0

Table of Contents

1	Introduction	1
1.1	Description of the problem	1
1.2	Purpose	1
1.2.1	Goals	2
1.3	Scope	2
1.4	Definitions, acronyms and abbreviations	2
1.5	Reference Documents	3
1.6	Overview	4
2	Overall Description	5
2.1	Product perspective	5
2.2	Product functions	5
2.3	User characteristics	5
2.4	Constraints	5
2.5	Assumptions	5
2.5.1	Text assumptions	5
2.5.2	Domain assumptions	5
3	Specific Requirements	6
3.1	External Interface Requirements	6
3.1.1	User Interfaces	6
3.1.2	Hardware Interfaces	6
3.1.3	Software Interfaces	6
3.1.4	Communication Interfaces	6
3.2	Functional Requirements	6
3.3	Non Functional Requirements	7
4	Scenarios	8
5	Modeling the Requirements	11
5.1	UML Models	11
5.1.1	Use Cases	11
5.1.2	Class Diagram	12
5.1.3	Sequence Diagram	12
5.1.4	Statechart Diagram	12
5.1.5	Activity Diagram	12
5.2	Alloy	12
5.2.1	Source Code	12
5.2.2	Generated World	19

6	RASD Preparation	22
6.1	Tools	22
6.2	Timing	22

List of Figures

5.1	Result of the execution of Alloy code	19
5.2	Generated world 1	20
5.3	Generated world 2	21

1 | Introduction

1.1 Description of the problem

The given problem consists in developing a system, *PowerEnJoy*, for managing a car-sharing service. This system will let the users register and get a personal account, search for available cars over the city, reserve and pick up cars. The system will be implemented as a web-application so it can be used from either mobile devices or personal computers. It is possible to identify two main types of people that will interact with *PowerEnJoy*, also called *actors*:

- Users
- Employees

1.2 Purpose

Analyzing the system-to-be in a more detailed way, it is possible to better define the functionalities that the system should provide to the users and the employees.

The users will have the possibility to register to get a personal account on *PowerEnJoy*, which is by the way mandatory to access to all the functionalities provided. Once the user is logged-in, he can search for an available car within a certain range from his current position or from a given address and, in case, he can pick up the car. He has also the possibility to reserve a chosen car for up to one hour before the ride (he can also delete the reservation subsequently if the car won't be used: in this way he is free from fees). The users, while registering, have to provide all personal information, a valid driving license and payment info; the whole procedure of the payment will rely on an external payment system, deferring to it all issues about information validity.

Moreover, to promote a more eco-friendly behaviour of the users, the system will apply special discounts or extra charge on the cost of the rides according to some different conditions.

For what concerns discounts, users that satisfy at least one of the following conditions will be awarded : the user takes at least two other passengers (-10%); the car is left with no more than of 50% empty battery(-20%); the car is left at special parking area and the user takes care of plugging the car into the power grid (-30%). Additionally, the system provides a special option, called *money saving option*, which lets the user insert his/her final destination and he/she gets from the system information about the area where to park the car in order to get a special discount. The area selected from the system ensures a uniform distribution of the car in the city and depends also on the final destination the user has inserted.

Talking about extra charges, the last ride of a user will be charged with a 30% more if the car is left at more than 3 km from the nearest charging area or with the battery level

under 20%.

Regarding the company employees, the system will let them use special functions to retrieve specific information about the cars (e.g.: battery level, internal working status) and their positions. In this way, they can move cars from safe areas to charging areas and viceversa if needed.

1.2.1 Goals

From the previous description of the *Purpose* of the system, it is possible to define the following goals that the system has to fulfil to catch the *User* needs.

- [G1] The user can apply the registration form
- [G2] The user can log in
- [G3] The user can search for a car in a certain location within a certain range
- [G4] The user can search for a car in his current position within a certain range
- [G5] The user can rent an available car on the spot
- [G6] The user can reserve a car
- [G7] The user can delete the reservation of a car
- [G8] The user can obtain special discounts or penalties on his last ride

Furthermore, since the system will be used by company employees too, it has to fulfil the following goals with respect to the *Employee* actor.

- [G9] The employee can log in into the system
- [G10] The employee can search for the cars over the city
- [G11] The employee can update the status of the cars

1.3 Scope

1.4 Definitions, acronyms and abbreviations

System: sometimes called also *system-to-be*, represents the application that will be described and implemented. In particular, its structure and implementation will be explained in the following documents. People that will use the car-sharing service will interact with it, via some interfaces, in order to complete some operations (e.g.: reservation and renting).

Renting: it is the act of picking-up an available car and of starting to drive.

Ride: the event of picking-up a car, driving through the city and parking it. Every Ride is associated to a single user and to a single car.

Reservation: it is the action of booking an available car. **Car:** a car is an electrical vehicle that will be used by a registered user.

Not Registered User: indicates a person who hasn't registered to the system yet; for

this reason he can't access to any of the offered function. The only possible action that he can carry out is the registration to get a personal account.

Registered User: interacts with the system to use the sharing service. He has an account (which contains personal information, driving license number and payment data) that must be used to access to the application in order to exploit all the functionalities.

Employee: it's a person who works for the company, whose main task is to plug into the power grid those cars that haven't been plugged in by the users. He is also in charge of taking care of the status of the cars and of moving the vehicles from a safe area to a charging area and vice versa if needed.

Safe Area: indicates a set of parking lots where the users have to leave the car at the end of the rent; the set of the Safe Areas is pre-defined by the system management. These areas are spread all over the city.

Plug: defines the electrical component that physically connects the car to the power grid.

Charging Area: is a special *Safe Area* that also provides a certain number of plugs that connect the cars to the power grid in order to recharge the battery.

Registration: the procedure that an unregistered user has to perform to become a registered user. At the end, the unregistered user will have an account. To complete this operation three different types of data are required: personal information, driving license number and payment info.

Search: this functionality lets the registered user search for available cars within a certain range from his/her current position or from a specified address.

RASD: is the acronym of *Requirements Analysis and Specification Document*

1.5 Reference Documents

During the writing of this document, the following resources have been taken into account:

- Specification Document:
 - *Assignments+AA+2016-2017.pdf*
- *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.*
- Example document:
 - *RASD sample from Oct.20 lecture.pdf*
- Papers on *Green Move Project*

1.6 Overview

In this first chapter, after a brief description of the given problem, the explanation of the system-to-be and its application domain (according to the *World and the Machine Paper*), the goals that the system have to achieve have been stated out. Subsequently, a glossary has been inserted in order to help the reader in identifying in a unique way some words and concepts, that will be used later on in this document and also in the next ones.

In the second chapter, there is a small presentation of the product perspective and the main functionalities; after that, a basic analysis and characterization of the beneficiaries (users and employees) of the system-to-be is pointed out. The last part of this chapter covers the assumptions that have been made in order to develop the system.

In the third chapter, which is the core of the RASD, the Functional Requirements are derived from the goals defined in the first chapter. Next, Non Functional Requirements will be presented.

The fourth chapter gathers a bunch of scenarios for all the possible usages of the system. In the fifth chapter, the Functional Requirements of the system-to-be are modelled firstly using a semi-formal notation (UML), via different types of diagrams, and secondly through Alloy code; in the end of the chapter the graphical output of the Alloy Analyser is given. The sixth chapter include the tools that have been used to redact this document and the time effort required to complete it.

2 | Overall Description

2.1 Product perspective

2.2 Product functions

2.3 User characteristics

2.4 Constraints

2.5 Assumptions

2.5.1 Text assumptions

- [TA1] Regarding to discounts awarding, in case many conditions are simultaneously satisfied, the system will reward the user with the maximum discount among the available ones
- [TA2] Regarding to discounts awarding, in order to get a discount of 30% for plugging the car into the power grid, the user has no more than 5 minutes to complete the action. This is a reasonable time to allow the user to get out of the car (and possibly to pick up the bags in the trunk) and correctly plug in it. Furthermore, this is also done to ensure that it's really the user that actually carries out the action and not someone else (e.g. the employee)
- [TA3] Regarding to the ride experience, to identify in a unique way the end of the rent, the user, just after shutting down the engine, has to push a button on the car screen to confirm the end of the ride

2.5.2 Domain assumptions

- [DA1] We assume the user enters valid payment credentials
- [DA2] The GPS position is accurate
- [DA3] The payment info of the user is correct
- [DA4] The payment is assigned to an external service
- [DA5] The system verifies that the remaining balance is positive
- [DA6] The number of passengers is known thanks to sensor measurements inside the car

3 | Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

3.1.3 Software Interfaces

3.1.4 Communication Interfaces

3.2 Functional Requirements

Deriving requirements from goals.

[G1] The user can apply the registration form

[R1] The system checks whether the user entered all the required data.

[R2] The system checks whether the current user is already registered or not.

[R3] The systems checks that there is only one user with that name.

[R4] The system generates a password for the access and sends it back to the new user.

[DA1]

[G2] The user can log in

[R5] The system checks the input data and verifies whether the user is already registered.

[R6] The system confirms the access.

[G3] The user can search for a car in a certain location within a certain range

[R7] The system verifies whether the given location exists.

[R8] The system shows the available cars within the range of distance specified by the user from the given location.

[DA2]

[G4] The user can search for a car in his current position within a certain range

[R9] The system obtains the user position via GPS.

[R10] The system shows the available cars within the range of distance specified by the user from his current position.

[DA2]

[G5] The user can rent an available car on the spot

[R11] The system obtains the user position via GPS and checks he is nearby.

[R12] The system verifies the car is available.

[R13] The system unlocks the car.

[DA3], [DA4]

[G6] The user can reserve a car

[R14] The system lets the user choose a car among those that are in the specified area.

[R15] The system checks that the car is available.

[R16] The system flags the car as Reserved.

[DA5]

[G7] The user can delete the reservation of a car

[R17] The system checks that the user has one active reservation.

[R18] The system lets the user delete his active reservation.

[G8] The user can obtain special discounts or penalties on his last ride

[R19] The system checks the position of the car at the end of the ride.

[R20] The system checks the battery charge level at the end of the ride.

[R21] The system checks whether the car is plugged into the power grid by the user at the end of the ride.

[R22] The system computes the discount (or penalty) considering the previous points.

[DA6]

3.3 Non Functional Requirements

4 | Scenarios

Scenario 1

Mario is a young man who wants to go to his girlfriend house. After evaluating all the possibilities, he ends up with reserving an electric car. Since it is the first time, he registers to the car sharing system, PowerEnJoy. After entering his credentials (name, Social Security Number, address, license number, email) and payment info, Mario chooses his login name. The system checks the correctness of the input data and sends Mario an email to complete the registration together with a password for the access.

Finally Mario is able to enter his personal page on his smartphone. He searches for a car nearby (he selects a maximum distance of 1 km from his current position) and he finds some available cars in a safe parking just 0,5 km far. That's good news! He texts his girlfriend Anna and picks up the car.

Scenario 2

It's a rainy Sunday in Milan and Bob, Tom and John are going to the stadium to see the football match. Bob knows that today there is a strike of all the public vehicles and so he suggests his friends picking up an electric car. The match is at 3 p.m. but the three friends want to be sure to arrive to the stadium in time, so they decide to reserve the car as soon as possible. They are going to leave at 2 p.m. and the system lets them do the reservation only from 1 p.m. on. Since only Tom has a car license and an account on PowerEnJoy, he applies for reservation. He accesses his personal page and enters his GPS position to look for a car near his house. Unfortunately no car is available. He then enters Bob's home address and luckily he finds one and eventually carries out the reservation. The system verifies that the remaining credit is positive, and as soon as checked it marks the car as "reserved".

The three friends reach the car and start their trip. As they arrive at the stadium, to their great surprise they find out that they are entitled to a discount as the car screen displays: "Congratulations, you have been accompanied by at least two people! You get a 10% discount on your ride!".

Scenario 3

It's Saturday afternoon and Anthony is willing to throw a party this evening. He needs to go shopping at the supermarket but his dad picked up his car this morning. He decides to reserve a car for the trip back home so that he can bring the shopping bags. Anthony is already registered on PowerEnJoy but he has never reserved a car before. He searches for a car near the supermarket and luckily finds one. He's very cautious and decides to read all the terms and conditions of the reservation. He finds out that the system reserves the car for up to one hour from the confirmation and after that time, in case he does not pick it up, he will be fined 1 EUR. Nonetheless, Anthony is self-confident and decides to confirm the reservation.

After leaving the market, Anthony accesses to PowerEnJoy with his smartphone to tell the system he's nearby. When he reaches the car, he sees that the car is ready to go. He then

opens the trunk and puts his shopping bags in. As soon as the engine ignites, he is impressed by the big display near the radio: firstly, on the screen he reads "Welcome Sir, PowerEnJoy wishes you a pleasant trip! Fare count starts right now" and as he drives through the streets the system shows him the current charge of the car, the current bill and a GPS interface where safe areas for parking are marked by red labels. As Anthony turns in Golgi street, the car also signals that there is a special charging station nearby and if he decides to stop there and plug the car into the power grid he will be awarded with a 30% discount on the ride. Unfortunately Anthony's home is not so close so he keeps driving for a while. After about 20 minutes, Anthony finally reaches a safe parking near home and decides to stop the car there. As he turns it off, the display shows a nice message: "Had a great trip? We hope so. The battery charge level is still high ($\geq 50\%$): you are entitled to a 20% discount on this ride!". After closing the trunk, Anthony checks that the system automatically blocks the car and then he goes home.

Scenario 4

It's a very hot summer day in Milan and James wants to go swimming. His mother Jane suggests him picking up an electric car to go there. James thinks it's a good idea and so he turns on his personal computer and goes to PowerEnJoy site. After registering, he reserves a car and searches for safe parking areas near the swimming pool. There is one really close to it. James then reaches the car, he puts his bag in and starts the car. The pool is pretty far and James knows that he will pay some extra money for that. As he arrives, the display on the car signals that the battery charge level is very low: "Battery is low (more than 80% empty). You are charged a 30% more on the last ride to compensate for the re-charge on site."

Scenario 5

Franklin is very happy today because it is his grandfather's George birthday and so he decides to rent an electric car to visit his grandparents. George lives together with Maria (Franklin's grandmother) in a country house, in the Milan suburbs. Unfortunately the house is in an isolated area and the PowerEnJoy web app signals that there is only a standard safe parking near George's house and that the closest charging station from it is 4 km far. Franklin knows that he will be charged a 30% more on his ride (as he has read in the terms and conditions on the site) but nevertheless he reserves a car.

As he arrives at the safe parking and turns the car off, the display of the car notifies him: "The closest power station is pretty far (more than 3 km). You are charged a 30% more on the last ride to compensate for the re-charge on site." Franklin is still happy anyway because he knows that his grandfather will appreciate his visit.

Scenario 6

Tom is a young cheapskate and after considering all the possibilities to reach the bus station he ends up with the decision of renting an electric car. He chooses PowerEnJoy because he has read of the money saving option that is available on its cars. After registering on the site and reserving a car near his home, he reaches the parking to pick it up. As he gets in the car and starts the engine, he searches for the money saving option on the big display. He's prompted to enter his destination and after a while the system shows some parking areas and power stations with the corresponding available discounts. Tom chooses a charging station near the gym as a parking and starts riding. As he arrives, he plugs the car in the power grid and he obtains a 30% discount on his ride.

Scenario 7

Harold is fit man who lives in Milan. He works as an employee for PowerEnJoy, an electric car-sharing company of the city, and everyday he rides his folding bike around the city to monitor the state of the cars. As he arrives in the SafeArea in Da Vinci street, he finds out that a car has a flat tire. What a pity! Harold takes his smartphone, goes to PowerEnJoy web site and enters in administrator mode. He signals the unavailability of the car, marking it as "Broken".

Scenario 8

It's 9 a.m and Jessica's work schedule has just started. She is an employee at PowerEnJoy, the most famous electric car-sharing company in Milan. She applied for this job because she does not like to work in an office all day long. Her main task is to take care of the cars in the parking areas. As she arrives at the PowerEnJoy SafeArea in Verdi street, she finds out that a car needs to be charged. She accesses to the PowerEnjoy site with her smartphone and enters in administrator mode in order to mark the car as "On Charge". Jessica then unfolds her bike and puts it in the trunk of the car. As she plugs the car in the power grid, a client arrives with another car, finishing the ride. The client is in a hurry and forgets to plug in the car, so Jessica takes care of it too.

5 | Modeling the Requirements

5.1 UML Models

5.1.1 Use Cases

Diagrams

Descriptions

Name: Registration. Actors: User not registered. Entry conditions: There are no entry conditions. Flow of events: - The user compiles the form with his personal data. - The user inserts his external payment account. - The user inserts his driving license number. - The user confirms the registration thanks to the mail sent by the system. Exit conditions: - The user has an account and the system recognizes him. Exceptions: - The external payment account doesn't exist. - The driving license number doesn't exist or is invalid. - Some of the personal data are not accepted (for example a small password). In any of these cases the system asks the user to reinsert correct data.

Name: Search for a car. Actors: User registered. Entry conditions: The user must be logged into the system. Flow of events: - The user goes to the search interface. - The user decides to search for a car manually navigating the map or specifying a place or automatically using the GPS location system. - The user selects the range within he wants to find the car. - The system shows the available cars and the near parking areas. Exit conditions: The user sees on the display a list of cars or a map with cars and parking area positions. Exceptions: - The location doesn't exist. - GPS is off. All of these cases are solved displaying an error to the user. For example: if there are no cars "NO CARS AVAILABLE", if the GPS is off "GPS IS OFF".

Name: Reserve a car. Actors: User registered. Entry conditions: User must be logged in and must have a selected car thanks to the previous search. Flow of events: - User confirms the reservation with a duration of 1 hour. Exit conditions: The user has reserved a car that he is be able to take in an hour. He can check the reservation in the "MANAGE RESERVATION" panel and possibly cancel it. This car is marked as reserved. Exceptions: - User's balance isn't positive. The system displays an error and brings him back to the initial page.

Name: Pick up a car. Actors: User registered. Entry conditions: User must be logged in. Flow of events: - User wants to unlock a car which is not reserved by someone else. - The system unlocks the car verifying the user GPS position and the user balance. - User enters into the car. Exit conditions: The user can enter and drive the car, starting by clicking "START" on the car display. Exceptions: - The system can't verify the user position because the user GPS is off. The system asks the user to turn the GPS on. - User has a non-positive balance. The system displays an error to and brings him back to the initial page.

Name: Stop driving. Actors: User. Entry conditions: User must be logged in and must have taken a car. Flow of events: - User powers off the car. - User notifies on the car-display that the ride is over. - The system displays the cost of the service showing possible discounts or extra charges. Exit conditions: User exits the car and pays the service(with an external system). Exceptions: - The user can't afford the ride. - The user doesn't click on the finish button on the car display. In both of these cases the external system payment is responsible.

5.1.2 Class Diagram

5.1.3 Sequence Diagram

5.1.4 Statechart Diagram

5.1.5 Activity Diagram

5.2 Alloy

5.2.1 Source Code

```
open util/boolean

//Define the car
sig Car{
  state: one State ,
  batteryLevel: Int ,
  peopleOnBoard: Int ,
  maxPeople: Int ,
  location: Position ,
  locked: Bool
}
{
  batteryLevel ≥ 0
  batteryLevel ≤ 100
  peopleOnBoard ≤ maxPeople
  maxPeople = 5
  peopleOnBoard ≥ 0
}

//Define the abstract state of a car
abstract sig State{ }

//Define specific states of a car
sig AvailableState extends State{ }
sig BrokenState extends State{ }
sig InUseState extends State{ }
sig ReservedState extends State{ }
sig OnChargeState extends State{ }

//Define a safe area
sig SafeArea{
```



```

    position: Position ,
    occupiedParkingLots: Int ,
    totalOutlets: Int
}
{
    occupiedParkingLots ≥ 0
    totalOutlets > 5 and totalOutlets < 20
    totalOutlets ≥ occupiedParkingLots
}

//Define a charging area
sig ChargingArea extends SafeArea{
    carsInCharge: set Car
}
{
    carsInCharge.state in OnChargeState
    #carsInCharge ≤ totalOutlets
    occupiedParkingLots = #carsInCharge
}

//Define a user
sig User{
    Use: lone Car ,
    reservation: lone Reservation
}
{
    //If a user is riding a car then he can not have a
    reservation and viceversa
    #carInUse > 0 ⇒ #reservation = 0
    #reservation > 0 ⇒ #carInUse = 0
    carInUse ≠ none ⇒ carInUse.state in InUseState
}

//Define a Reservation
sig Reservation{
    user: User ,
    car: Car
}
{
    car.state in ReservedState
}

//Define a Ride
sig Ride{
    user: User ,
    car: Car ,
    time: Int ,
    cost: Int ,

```

```

    moneySavingOption: Bool,
    initialPosition: Position,
    finalPosition: Position,
    pluggedIn: Bool,
    discount: Int,
    extraCharge: Int
}
{
    time > 0
    cost > 0
    cost = computeCost[1,time]
    discount ≥ 0
    extraCharge ≥ 0
    initialPosition ≠ finalPosition
    car.state in InUseState
    initialPosition in (SafeArea.position + ChargingArea.
        position)
    finalPosition in (SafeArea.position + ChargingArea.position
        )
    car.batteryLevel = computeBatteryLevel[time]
}

//Define a position
sig Position{
    lat: Int,
    long: Int
}
{
    lat > 0
    lat < 8
    long > 0
    long < 8
}

/** FACTS */

// Each car can be used exclusively from a single user at a
// time
fact {
    all disjoint u1, u2: User | u1.carInUse & u2.carInUse =
        none
}

// Each reservation has a unique car
fact{
    all r1,r2: Reservation | r1 & r2 = none ==> r1.car & r2.car =
        none
}

//There can't be multiple users with the same reservation
fact{

```

```

    all u1:User| (u1.reservation ≠ none) ⇒ ( no u2:User| (u2
        & u1 = none) and u1.reservation = u2.reservation)
}
//Each car in use is assigned to only one ride
fact{
    all disjoint r1,r2:Ride| r1.car & r2.car = none
}
//Each user is either on a ride or reserves a car
fact{
    Ride.user & Reservation.user = none
}
//If the car is reserved or in available state then its charge
    is at 100\%
fact{
    all c:Car|(c.state in ReservedState or c.state in
        AvailableState) ⇒ c.batteryLevel = 100
}
//If the car is InUseState it must have at least one person on
    board
fact{
    all c:Car|(c.state in InUseState) ⇔ c.peopleOnBoard > 0
}
//If the car is InUseState, its charge is different than 0 and
    100 (Constraint added for readability)
fact{
    all c:Car|(c.state in InUseState) ⇒ c.batteryLevel ≠ 100
        and c.batteryLevel ≠ 0
}
//If a car is on charge its batteryLevel is less than 100 (
    Constraint added for readability)
fact{
    all c:Car|(c.state in OnChargeState) ⇒ c.batteryLevel <
        100
}
//Cars are always locked unless they're inUseState
fact{
    all c:Car|(c.locked = False) ⇔ c.state in InUseState
}
//Cars that are OnChargeState are in a chargingArea
fact{
    no c:Car|(c.state in OnChargeState) and c.location not in
        ChargingArea.position
}
//All cars not riding are in a safe area or in a charging area
fact{
    all c:Car|(c.state not in InUseState) ⇔ ( c.location in
        SafeArea.position or c.location in ChargingArea.position
    )
}

```

```

//All cars inUseState are assigned to a ride
fact{
  all c: Car|(c.state in InUseState) ==> c in Ride.car
}
//A user on ride can park in a SafeArea only if there is at
  least one free parking lot
fact{
  all r: Ride,s: SafeArea|(r.finalPosition in s.position ) ==>
    (s.totalOutlets > s.occupiedParkingLots)
}
//A user on ride can park in a ChargingArea only if there is
  at least one free parking lot
fact{
  all r: Ride,c: ChargingArea|(r.finalPosition in c.position )
    ==> (c.totalOutlets > c.occupiedParkingLots)
}
//A user that plugs the car in the power grid after a ride is
  in ChargingArea
fact{
  all r: Ride|(r.pluggedIn = True) ==> r.finalPosition in
    ChargingArea.position
}
//If user has a discount he can not have an extraCharge
fact{
  all r: Ride|(r.discount != 0) ==> (r.extraCharge = 0)
}
//If user has an extraCharge he can not have a discount
fact{
  all r: Ride|(r.extraCharge != 0) ==> (r.discount = 0)
}
//A user gets a 10\% discount on the last ride if and only if
  he's accompanied by at the least two people
fact{
  all r: Ride|(r.car.peopleOnBoard > 2 and r.car.batteryLevel
    < 50 and r.pluggedIn = False ) <==> r.discount = 10
}
//A user gets a 20\% discount on the last ride if and only if
  the batteryLevel is greater than 50\% at the end of the
  ride
fact{
  all r: Ride|(r.car.batteryLevel > 50 and r.pluggedIn = False
    ) <==> r.discount = 20
}
//A user gets a 30\% discount on the last ride if takes care
  of plugging the car in the power grid at the end of the
  ride
fact{
  all r: Ride|(r.pluggedIn = True) <==> r.discount = 30
}

```

```

//A user gets a 30\% discount on the last ride and the system
//guarantees a uniform distribution of cars among the
//SafeAreas if the user activates the MoneySavingOption
fact{
  all r: Ride | (r.moneySavingOption = True) ==> r.discount = 30
    and (all disjoint s1,s2: SafeArea | s1.
      occupiedParkingLots = s2.occupiedParkingLots)
}
//A user gets a 30\% extraCharge on the last ride if the
//batteryLevel is lower than 20\% at the end of the ride
fact{
  all r: Ride | (r.car.batteryLevel < 20) ==> r.extraCharge = 30
}
//A user gets a 30\% extraCharge on the last ride if he parks
//the car in a spot which is more than 3 km far from the
//nearest ChargingArea
fact{
  all r: Ride | all c: ChargingArea | distance[r.finalPosition,c.
    position,3] = True ==> r.extraCharge = 30
}
//The sets of possible values for the discount and extraCharge
//are finite
fact{
  all r: Ride | (r.discount = 10 or r.discount = 20 or r.
    discount = 30 or r.discount = 0) and (r.extraCharge = 0
    or r.extraCharge = 30)
}
//Each ride has only one user
fact{
  all r1: Ride,r2: Ride | (r1 & r2 = none) ==> (r1.user & r2.
    user = none)
}
//Each car is either on a ride or reserved
fact{
  Ride.car & Reservation.car = none
}
//For each reservation, the user of the reserved car is whom
//reserved the car
fact{
  all r: Reservation | r.user.reservation = r
}
//Each car must have a state
fact{
  all s: State | (s != none) ==> (some c: Car | c.state = s)
}

//This function returns true if the distance between the two
//given positions is greater than range, else false

```

```

fun distance(p1: Position, p2: Position, range: Int): Bool{
  (add[mul[minus[p1.lat, p2.lat], minus[p1.lat, p2.lat]], mul[
    minus[p1.long, p2.long], minus[p1.long, p2.long]]) > mul[
    range, range] implies True else False
}
//This function returns the cost of a ride given the fare and
//the time
fun computeCost(fare: Int, time: Int): Int{
  mul[fare, time]
}
//This function returns the batteryLevel of a car given the
//time of the ride
fun computeBatteryLevel(time: Int): Int{
  minus[100, mul[time, 1]]
}

//Each user who is riding a car can not have a reservation at
//the same time
assert RideOrReserve{
  all u: User | (u.carInUse ≠ none) ⇒ (no r: Reservation | r.
    user = u)
}
//If a car is broken then it can not be riding
assert BrokenCarNotRiding{
  all c: Car | (c.state in BrokenState) ⇒ (no r: Ride | r.car =
    c)
}
//If a user takes care of plugging the car in the power grid
//he is rewarded with 30\% discount on the last ride
assert PluggingTheCarIsGood{
  all r: Ride | (r.pluggedIn = True) ⇒ r.discount = 30
}

pred showWorld{
  #User > 1
  #Ride > 1
  #Reservation > 1
  #Car > 1
  #ChargingArea > 1
}

run showWorld for 5 but 8 int

check RideOrReserve for 8 int
check BrokenCarNotRiding for 8 int
check PluggingTheCarIsGood for 8 int

```

5.2.2 Generated World

Executing "Run showWorld for 5 but 8 int"

Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
523514 vars. 14470 primary vars. 1753077 clauses. 2751ms.

Instance found. Predicate is consistent. 14617ms.

Executing "Check RideOrReserve for 8 int"

Solver=sat4j Bitwidth=8 MaxSeq=4 SkolemDepth=1 Symmetry=20
292738 vars. 8616 primary vars. 997847 clauses. 1361ms.

• No counterexample found. Assertion may be valid. 281ms.

Executing "Check BrokenCarNotRiding for 8 int"

Solver=sat4j Bitwidth=8 MaxSeq=4 SkolemDepth=1 Symmetry=20
292741 vars. 8616 primary vars. 997849 clauses. 1303ms.

No counterexample found. Assertion may be valid. 135ms.

Executing "Check PluggingTheCarIsGood for 8 int"

Solver=sat4j Bitwidth=8 MaxSeq=4 SkolemDepth=1 Symmetry=20
293717 vars. 8613 primary vars. 999547 clauses. 1312ms.

No counterexample found. Assertion may be valid. 346ms.

4 commands were executed. The results are:

- #1: **Instance found.** showWorld is consistent.
- #2: No counterexample found. RideOrReserve may be valid.
- #3: No counterexample found. BrokenCarNotRiding may be valid.
- #4: No counterexample found. PluggingTheCarIsGood may be valid.

Figure 5.1: Result of the execution of Alloy code

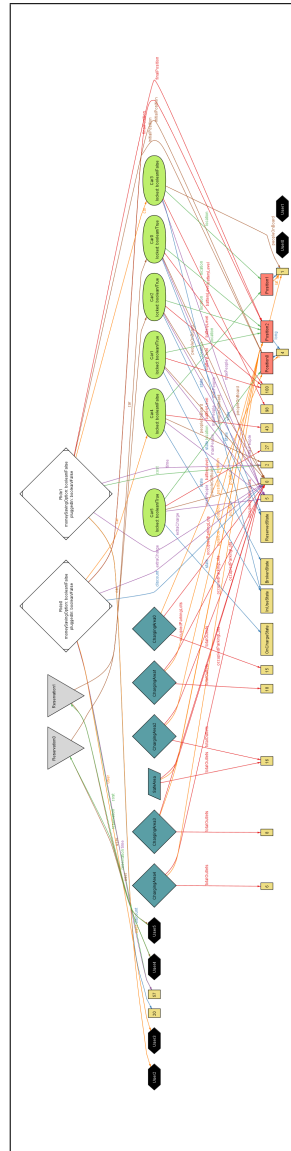


Figure 5.2: Generated world 1

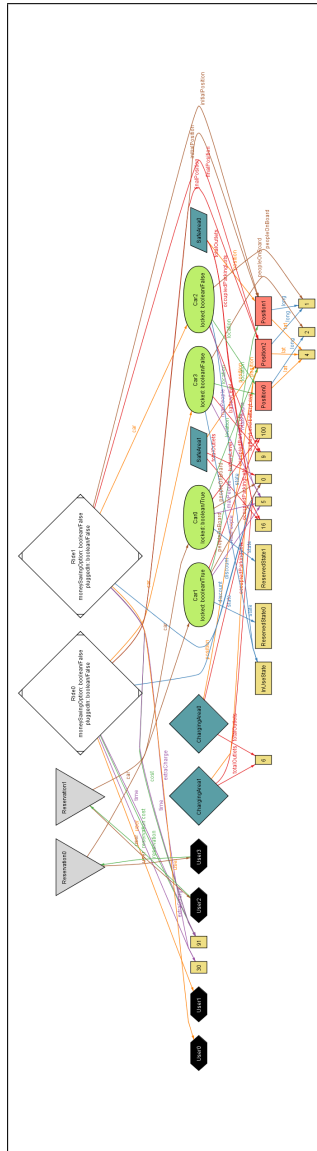


Figure 5.3: Generated world 2

6 | RASD Preparation

6.1 Tools

During the writing of this document, the following application tools have been used:

- Star UML, for creating all types of UML models for the requirements
- TeXstudio, for writing the document in \LaTeX
- AlloyAnalyzer, for running the Alloy code

6.2 Timing

[TBD]