Politecnico di Milano, A.Y. 2016/2017
M.Sc. Degree Programme in Computer Science and Engineering
Software Engineering 2 Project

Requirement Analysis and Specification Document

Philippe Scorsolini,
Lorenzo Semeria,
Gabriele Vanoni

January 4, 2017

Contents

1. Introduction

PowerEnJoy is a car-sharing service that exclusively employs electric cars.
Users must have an account on the online platform to be able to access the service. It is possible via the mobile app or directly via a web browser to look for available cars within a certain distance from user current location (possibly tracked by GPS) or from a specified address. Once the user selects a car he can book it for up to one hour, if he does not pick it up he is fined and his reservations is voided. Rates are calculated based on time spent driving and cars must be parked in predefined safe areas, some of which are equipped with power grids.
The management system encourages eco sustainability and best practices with a *bonus-malus* method. For examples there are discounts for car pooling and extra charges for users that park far from power grid stations or leave the battery very low.

1.1. Purpose

This document is the Requirement Analysis and Specification Document (RASD). Its main objective is to accurately describe the system, in terms of both functional and non-functional requirements. Moreover it focuses on modeling the system taking into account the customer's needs, as well as showing the constraints and limits that must be imposed and the most relevant use cases. This document is targeted to anyone who has to contribute to PowerEnjoy since developers will use it to check the requirements that need to be implemented and it may be used as a contract (or part of it) between the developer and the customer.

1.2. Scope

The aim of the project is to create a brand new system that allows customers to access the car sharing service offered by PowerEnJoy. The system will let the customers register and then login, managing the sensible data submitted during the registration and checking automatically or via an operator the correctness of the information. The system will allow logged customers to reserve available cars both via web or mobile app, access the cars once they reach them via mobile app using positioning data and then compute and apply the fare once they park them in predetermined "safe zones", applying discounts depending on whether or not they followed some good practices. The system will give an interface to the system administrator in order to manage the cars' pool, accessing their status and position and will also notify an operator when a car needs to be moved from where it has been parked. In the event of an accident an operator will be notified and will reach the damaged car, taking care of it and reporting the car's status to the system.

1.3. Abbreviations, Definitions and Acronyms

1.3.1. Abbreviations:

- Gn: the n-th Goal
- An: the n-th Assumption
- Rn: the n-th Requirement

1.3.2. Acronyms

- CC: Credit Card
- DL: Driving Licence
- AU: Active User

1.3.3. Definitions

- Visitor: person that may not be registered to the system or not logged in.
- User: a registered and logged in Visitor, that may be still waiting for his information to be verified.
- Active User: a User whose data (CC, DL) have been verified. (Shares all User's characteristics)
- Safe Zone: predefined zones where parking is allowed, parking is forbidden in any other zone.
- Park: park the car in the safe zone and terminate the rental.

1.4. Actors

- Visitor: see definition
- User: see definition
- Active User: see definition
- System Administrator: person in charge of the maintenance of the system and the management of cars' information
- Operator: person that physically actuates any operation needed by the system in the physical world
- External geographical database

1.5. Goals

PowerEnJoy should provide these functionalities:

G1) Allow visitors to sign up.

G2) Allow visitors to log in.

G3) Allow Users and Active Users to update or modify their profile's information.

G4) Show updated information on available cars.

G5) Allow Active Users to reserve a car.

G6) Allow Active Users to unlock the car reserved

G7) Compute the fare.

G8) Allow System Administrator(s) to update system's information.

G9) Ensure that the fare is paid.

G10) Allow the driver to choose the money saving option and get near their destionation.

G11) Allow the user to park the rented car in safe zone.

1.6. Reference documents

- [SDA16] Specification Document: "Assignments AA 2016-2017.pdf"
- [IEEE11] ISO/IEC/ IEEE 29148:2011(E) "Systems and software engineering — Life cycle processes — Requirements engineering"
- Given examples:
  - "RASD sample from Oct. 20 lecture.pdf"
- [Jack12] Daniel Jackson, *Software Abstractions: Logic, Language, Analysis*, MIT Press, 2012

2. Overall description

2.1. Product perspective

The future product will consist of a mobile application and a web interface for costumers and an internal interface for the system administrator(s). A mobile phone with the app installed will be necessary in order to unlock the reserved cars, whereas it will be possible to reserve cars through both the mobile application and the web interface, after logging in.

2.2. User characteristics

Everyone with a mobile phone can download and install the app and everyone with an internet connection can access the web app but only visitors with a valid DL can create an account.

2.3. Constraints

2.3.1. Regulatory policies

The system must require users the permission to access their position data and has to manage sensible data (position, phone number, email, DL, CC) respecting the privacy laws in force in the current country where the system is being deployed.

2.3.2. Hardware limitations

The following are mandatory requisites for the hardware in order to access the system via the different interfaces:

- Mobile app:
  - data connection
  - positioning services (GPS, Cell-ID, Wi-Fi)
  - enough storage space to install the package
- Web app:
  - web browser
  - internet connection

2.4. Assumptions and Dependencies

A1) There is at least one System Administrator that manages the website and the app.

A2) The System Administrator can add and remove cars from the system and manage their status if needed.

A3) There are operators that can recover cars.

A4) Only Active Users can use the service.

A5) Every car has a unique code (different from the licence number) that is visible from the outside. This number is used to ensure that a User is near a Car if he tries to unlock it and no GPS data is available.

A6) The licence Office (Motorizzazione Civile) allows for automated check of Licence Numbers, also providing checks for Name and Surname (at least). This check can be done in any moment of the day and gives immediate feedback.

A7) Foreign licences are allowed, but will be checked personally by an operator.

A8) An active user can reserve up to one car at any given time.

A9) Every car has a screen.

A10) Every car is connected to the internet and can change the displayed information, which can be remotely updated by the system.

A11) As soon as an Active User's DL o CC expires, he is downgraded to User (cannot book/rent cars) until he updates the relevant information and is notified of the problem.

A12) An AU can temporarily switch the engine off ("park" the car in the common meaning) and keep paying for the rental in order to be able to pick up the same car again. This is only possible outside the Safe Zone.

A13) An Assistance Service via phone will be provided and will deal with all the phenomena the system can not handle (e.g. accidents, paying fees and any other problem with cars not explicitly indicated in this document as held by the system).

3. Specific Requirements

3.1. External Interface Requirements

PowerEnjoy makes mainly use of two external providers, one for the maps and one to handle payments. The map provider's API will be used only to download location-specific maps, over which the car's position will be shown as an overlay. The payment handler will have to take care of the transaction, ensuring a suitable secure connection between the user and the service. Moreover the API will have to provide information about the outcome of a transaction, wether it succeeded or it failed and the reason for the failure. Furthermore, PowerEnjoy will use the Licence Office (Motorizzazione Civile) public API to verify Driving Licences and retrieve driver's informations in order to autametically accept local users.
Both the web application and the mobile application will access the same APIs offered by PowerEnjoy in order to login and retrieve information.

3.2. Functional Requirements

3.2.1. G1: Allow visitors to sign up

R1) A valid email is required during the registration.

R2) A valid DL is required.

R3) Invalid emails are all addresses that are either already in use or not well formed.

R4) Email Address must be verified.

R5) The DL is valid if all the fields match the user's information and the Licence Office confirms that the Licence Number, the Name and the Surname are correct.

R6) Foreign DL must be verified by an Operator.

R7) If some of the provided information is incorrect, the registration is rejected and the user is prompted to fix the issue(s).

3.2.2. G2: Allow visitors to log in

R8) A previously registered email is required.

R9) The password chosen during the registration of the submitted email has to be inserted in order to log in.

R10) Visitors submitting incorrect email and/or password shall not be allowed to log in.

3.2.3. G3: Allow Users to update or modify their profile's information

R11) Every registered User can modify his information.

R12) Changing email address is allowed but the new email must be confirmed for the account to be valid.

R13) Changing the CC information is allowed as long as the new CC is valid.

R14) Changing the DL is allowed but the Licence has to be verified.

R15) Changing name and/or surname is not allowed.

3.2.4. G4: Show updated information on available cars

R16) The list of available cars always includes only cars that are parked and not reserved and each is shown on a map in the location where it is actually parked.

R17) If a car is reserved is tagged on the map as reserved.

R18) For every car is displayed the remaining percentage of the battery.

R19) Users should be able to apply filters to show only cars within a certain distance from a specified location or with a minimum percentage of battery left.

3.2.5. G5: Allow Active Users to reserve a car

R20) Only Active Users can reserve cars.

R21) Cars can be reserved by a specific user for up to one hour before being unlocked.

R22) If a reserved car is not unlocked in one hour the reservation expires and the user pays a fine of 1€.

R23) Only available cars can be reserved.

R24) If an active user is already using a car he cannot reserve another one.

R25) The system shall provide the user the possibility to delete his pending reservation.

R26) The deletion of a reservation shall be allowed only before the reserved car has been unlocked.

3.2.6. G6: Allow Active Users to unlock the car they reserved

R27) A car can be unlocked only by the user who has reserved it.

R28) There exists a mechanism of acknowledgment between the car and the user.

R29) The car is unlocked only after the user is acknowledged.

R30) If a user unlocks the car without igniting the engine, the systems starts charging the regular price after the pick up time (one hour from the reservation) expires.

R31) If a user does not ignite the car within 15 minutes from the moment he unlocks it, the systems prompts the user to confirm he is fine. If no answer is received, an operator checks the car.

3.2.7. G7: Compute the fare

R32) The fare takes into account all price modifiers (discounts or extra fees).

R33) The computed fare is based on how many minutes have passed since the engine was ignited.

R34) If the engine was never ignited the fare is calculated as explained in R30.

R35) The computed fare is shown real-time on the car's screen.

R36) The system shall be able to apply predetermined discounts to users following some predetermined behavior.

R37) The system shall be able to know the number of passengers into a specific rented car.

R38) The system shall be able to know the battery percentage of a specific car in any moment.

R39) The system shall be able to locate a car and know whether or not it is plugged at one of the predetermined parking areas in any moment.

R40) The system shall be able to measure the distance of a specific car from the nearest power grid station.

R41) The system shall be able to make an Operator charge a specific car.

R42) The system shall know if a user selected the money saving option.

3.2.8. G8: Allow System Administrator(s) to update information

R43) The System Administrator is granted the necessary permissions allowing him to access each cars' data, status and position.

R44) The system shall present an Interface for the System Administrator in order to access with the necessary permissions.

R45) Only those accessing the systems with these permissions shall be able to access sensible data regarding the cars.

R46) The system shall be able to check the consistency of the information modified given a set of rules.

3.2.9. G9: Ensure that the fare is paid

R47) As soon as the rental is ended, use the payment method provided by the AU to pay the fare.

R48) If the provided method is invalid (e.g. expired or empty CC), notify the user.

R49) Users with unsuccessful pending fare shall not be allowed to book cars.

R50) Allow users to specify the main payment method.

R51) Periodically try to charge pending fare through the specified main payment method.

3.2.10. G10: Allow the driver to choose the money saving option

R52) The systems suggests a charging station near the user's destination if present (within a user-selectable radius).

R53) The suggested station must have at least a free plug to be suggested.

R54) The suggested station shall be determined to ensure a uniform distribution of cars.

R55) If the AU doesn't park in the suggested station the money saving option shall not be taken into account for future decisions.

R56) If the AU chooses this option, the system gives him the address and shows directions on the smartphone.

3.2.11. G11: Allow the user to park the rented car in safe zone

R57) Parking shall be allowed only in safe zones.

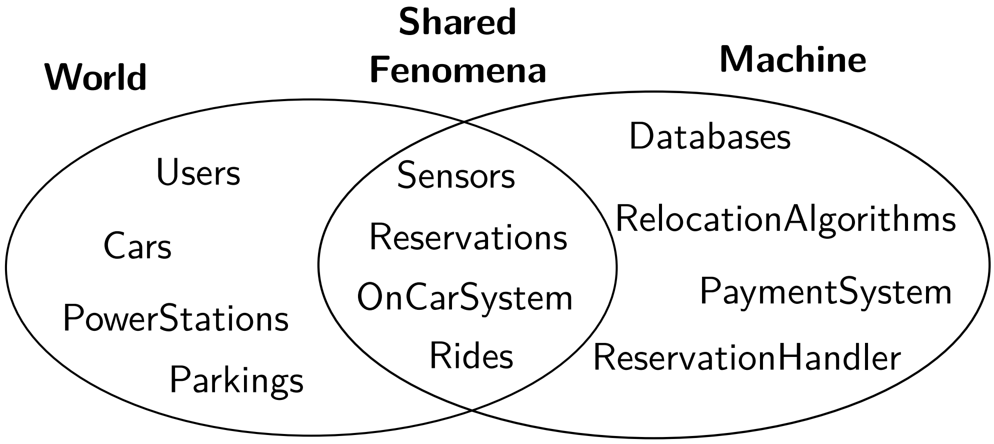
R58) The system shall not charge the user after the car has been parked and he has exited the car.

R59) The system shall lock the car, if parked in a safe area and the user has exited the car.

R60) The system shall be able to recognize whether or not there is a user inside the car.

3.3. The world and the machine

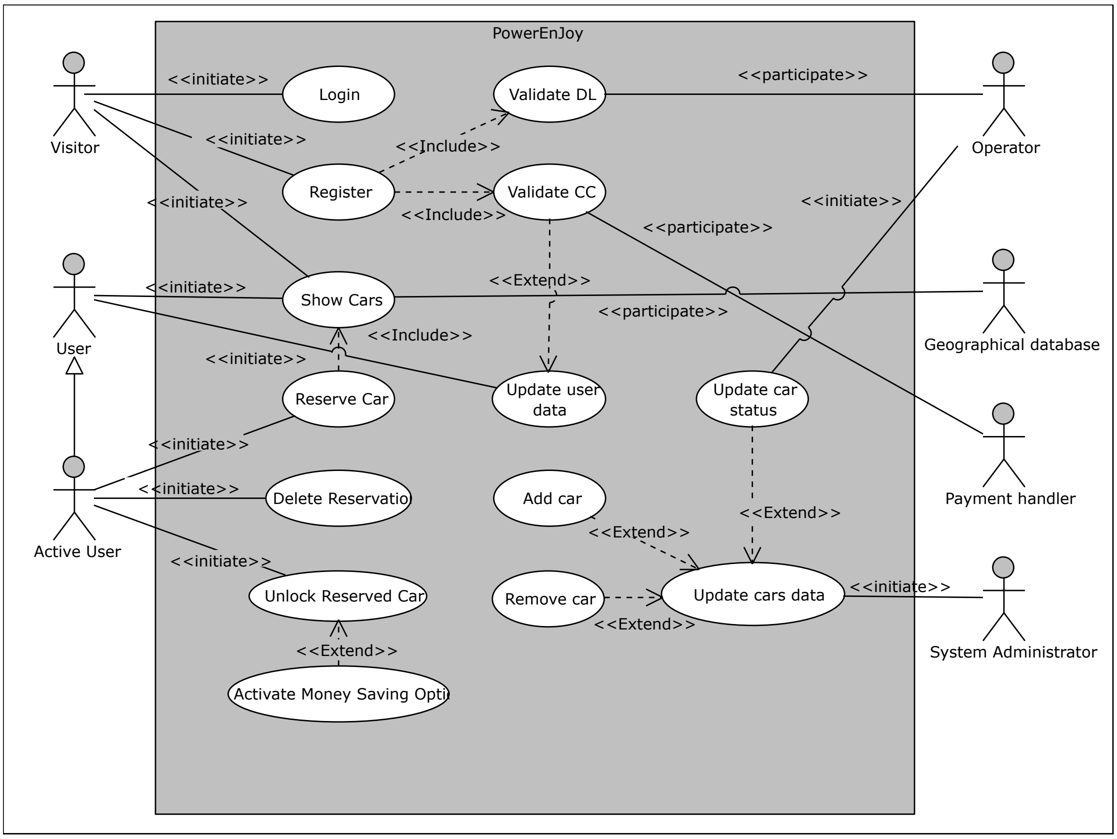
We developed an analysis about "the world and the machine" according to the paper [1]. We used the form of Venn diagrams.



[1] M. Jackson and P. Zave, *Deriving specifications from requirements: an example*, Proceeding ICSE '95, p. 15-24

3.4. Scenarios

3.4.1. Use case diagram



3.4.1.1. Visitor sign up

- Actor : Visitor
- Goal : G1 Allow visitors to sign up
- Entry conditions : None
- Flow of events :
  1. Opens the PowerEnJoy web page or mobile app.
  2. Clicks on the “Sign Up” button.
  3. He is asked to input a valid email address and a password.
  4. The system checks whether or not the email and the password submitted are valid.
  5. The visitor is asked to insert his driving licence and payment method data.
  6. If the data is correct the system stores the submitted data.
  7. The visitor will be logged in automatically and redirected to his account page.
  8. The user will be asked to confirm his email through a confirmation link the system sent him.

- Output conditions: The visitor completes the registration and becomes a User, can now on log in using the submitted email and password and shall wait for his DL to be confirmed by the system in order to become an Active User.
- Exceptions :
  1. If the inserted email is already in use or not valid the visitor will be prompted to log in or use a different email.

3.4.1.2. Visitor log in

- Actor : Visitor
- Goal : G2 Allow visitors to log in
- Entry conditions : the Visitor shall already have signed up
- Flow of events :
  1. Opens the PowerEnJoy web page or mobile app.
  2. Clicks on the "Log in" button.
  3. He is asked to input the registered email and the chosen password.
  4. The system checks if the submitted data are correct.
  5. Only if the data are correct the user is redirected to his account page.
- Output conditions : The visitor is logged in and become a User or Active User if his DL has been already confirmed.
- Exceptions :
  1. If the submitted email or password are incorrect the user will be asked to modify them and try again.

3.4.1.3. See rentable cars

- Actor : Visitor
- Goal : G4 Show updated information on available cars
- Entry conditions : None
- Flow of events :
  1. Opens PowerEnJoy web page or mobile app.
  2. A map in the home screen shows the position of available cars.
  3. A click on an available car shows the battery percentage of the car.
- Output conditions : None
- Exceptions : None

3.4.1.4. Reserve a car

- Actor : Active User
- Goal : G5 Allow Active Users to reserve a car
- Entry conditions : None
- Flow of events :
  1. Opens PowerEnJoy web page or mobile app.
  2. A map in the home screen shows the position of available cars.
  3. A click on an available car shows the battery percentage of the car and prompts the user to rent the selected car.
  4. Clicking on the "Reserve Now" button the user will rent the car.
  5. The user will be redirected to the reservations page, where he can see the history of his rents and the countdown for the current reservation.
- Output conditions : the user has reserved a car and has an hour to reach the car
- Exception : None

3.4.1.5. Delete a reservation

- Actor : Active User
- Goal : G5 Allow Active Users to reserve a car
- Entry conditions : the user shall have reserved a car and his reservation shall not be already expired
- Flow of events :
  1. Opens PowerEnJoy web page or mobile app.
  2. Goes to the reservations page.
  3. Clicks on the current reservation.
  4. He will be asked whether or not he wants to delete his reservation.
  5. Chooses to delete his reservation.
  6. The reservation is deleted and the user is redirected to his reservations page.
- Output conditions : the reservation has been cancelled and the user has no active reservation
- Exception : None

3.4.1.6. Unlock reserved car

- Actor : Active User
- Goal : G6 Allow Active Users to unlock the car they reserved
- Entry conditions : The user shall have reserved a car and his reservation shall not be already expired.

- Flow of events :
  1. Opens PowerEnJoy mobile app.
  2. Goes to the reservations page.
  3. Clicks on the current reservation.
  4. He will be asked whether or not he want to unlock the car.
  5. Chooses to unlock the car he reserved.
  6. The car is unlocked.
  7. The user can enter the car and ignite the car.
- Output conditions : the user has unlocked the car and can now use it.
- Exception : The user is not recognized as nearby so the car is not unlocked

3.4.1.7. Update cars' information

- Actor : System administrator
- Goal : G8 Allow System Administrator(s) to update information
- Entry conditions : None
- Flow of events :
  1. Opens PowerEnJoy system administrator interface.
  2. Goes to the cars data management page.
  3. Searches for the desired car which information should be updated.
  4. Modifies the car information as needed.
  5. Clicks the "Save" button.
  6. The system checks if there are no inconsistency introduced by the update.
  7. The system updates the status of the car and acts accordingly.
  8. The System administrator is redirected to the data management page.
  9. He clicks exit.
  10. He is redirected to the main page of the system administrator interface.
- Output conditions : the car's data have been successfully updated
- Exception : if the update introduces some inconsistency in the data the system administrator is prompted to correct the incorrect information.

3.4.2. Visitor interaction with the system

The interaction between the visitor and the system is modelled with UML Activity Diagram (in fig. 1) and BPMN notation (in fig. 2).

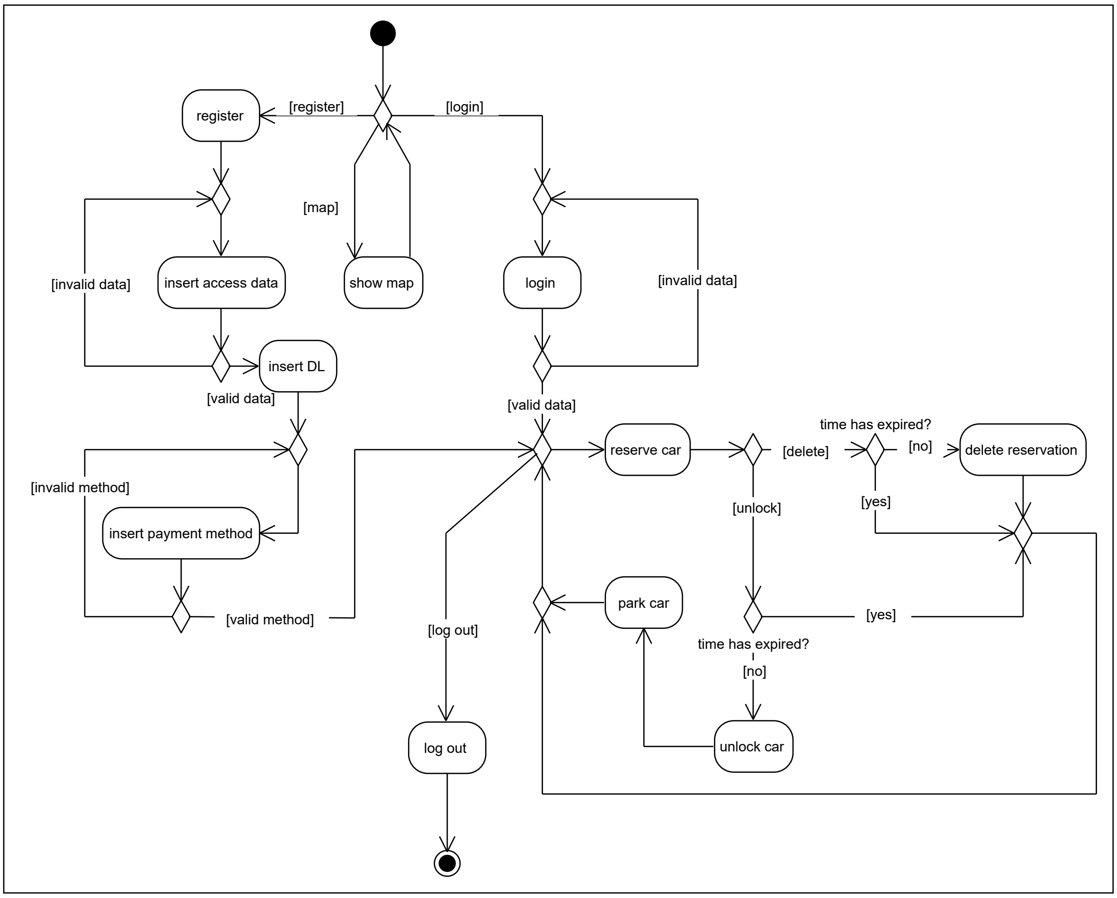


Figure 1: UML Activity diagram

3.4.3. Michelle needs to go to the train station:

Michelle needs to go to the train station but, not being on a hurry and wanting to save some money for her upcomping journey, she decides to use the “money saving option” as well as sharing the trip with two of her friends. She inputs her final destination and selects a radius of 3.5km from it. The system decides what the best charging station within that distance is and shows its address and directions to reach it. Michelle starts driving but unfortunately she is caught in a traffic jam. She opts to park closer to the train station fearing that she might miss her train. When she finally gets close to the station she parks the car and she and her friends leave. The system detects that they are not in the proposed charging station and doesn’t apply the discount for the “money saving option”.

3.4.4. Jack wants to register

Jack is an Erasmus student who just came into town. Knowing that he may need a car from time to time he decides to register on the website immediately in order to be able to use PowerEnjoy whenever he needs it. He registers on the website, providing his personal details as well as a valid email. He mistakenly inserts his email with a quotation mark instead of an at sign. The registration page notifies him and he fixes the mistake. He then confirms his email address by opening the email he received. Having done this he is

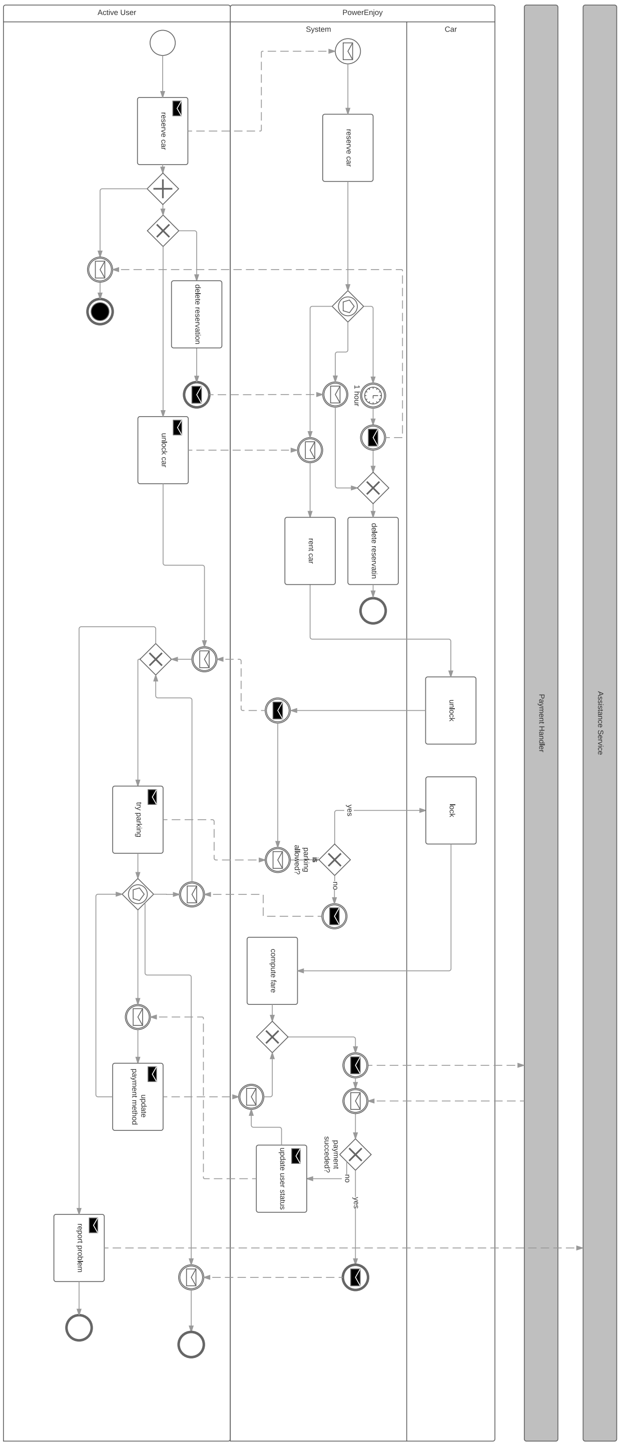


Figure 2: BPMN diagram representing the process of reservation of a car

considered a “Registered User” by the system, but he still needs to input his DL and payment method. Being a foreigner he provides an image of both sides of his DL as well as his Credit Card details. He then waits for the confirmation, since an Operator must check it. Once his Licence is confirmed he can book cars and use the system.

3.4.5. Francis forgot that his prepaid card is empty

Francis has picked a car up to go shopping with his Fiancé. When they are done with shopping they go back home and he parks the vehicle in a parking space nearby. Once he gets out of the car and locks it, he is notified that the payment was unsuccessful. He is also informed that until he pays for that rental he won’t be allowed to use the service. He then remembers that he forgot to load the prepaid card he is using for PowerEnjoy. As soon as he gets home he loads the card from his Home Banking and asks the app to try again with the payment. The transaction is accepted and Francis is allowed to book other cars whenever he pleases.

3.4.6. Shawn has trouble parking

Shawn wants to park his car but the system will not let him, pretending that he is not inside a safe zone. Since he is certain that he is inside a safe zone he contacts the Company to report the issue hoping that he will be allowed to leave the car since he has no time to spare. The Operator handles the problem as per company policy trying to help Shawn as much as he can.

3.4.7. Claire’s car broke down

Claire is riding on the car she rented with no issues in the city center and is speeding towards her sister’s country house. After having left the city her car suddenly lights up a warning light and she pulls over to check. Hoping that it will not endanger her life she decides to keep going but the car refuses to start. Quite worried she calls PowerEnjoy’s helpline which handles her problem as per company’s policy.

3.4.8. Matthew’s licence expires

Matthew has been using PowerEnjoy for quite some time and is very happy with the service since he never had any problem. One morning, being late for work, he decides to use PowerEnjoy to save some time. To his surprise, he is not allowed to rent the car just around the corner. From the app he is notified that his DL is expired and that he will not be allowed to use the service until he updates his DL information. Luckily enough, he has already renewed his Licence. He opens his profile from the App and proceeds with the information update procedure. He is compelled to update his DL only, since the other informations are still valid. He inputs his DL details and hopes that the verification will be fast. Since he has got a non-foreign licence, the process is completely automated and he manages to get on the car after a few minutes.

3.5. Non functional requirements

Althought in [A16] there is no reference in terms of quality of service, as for any other software artifact we can imagine that the system should behave within some quality constraints. In particular:

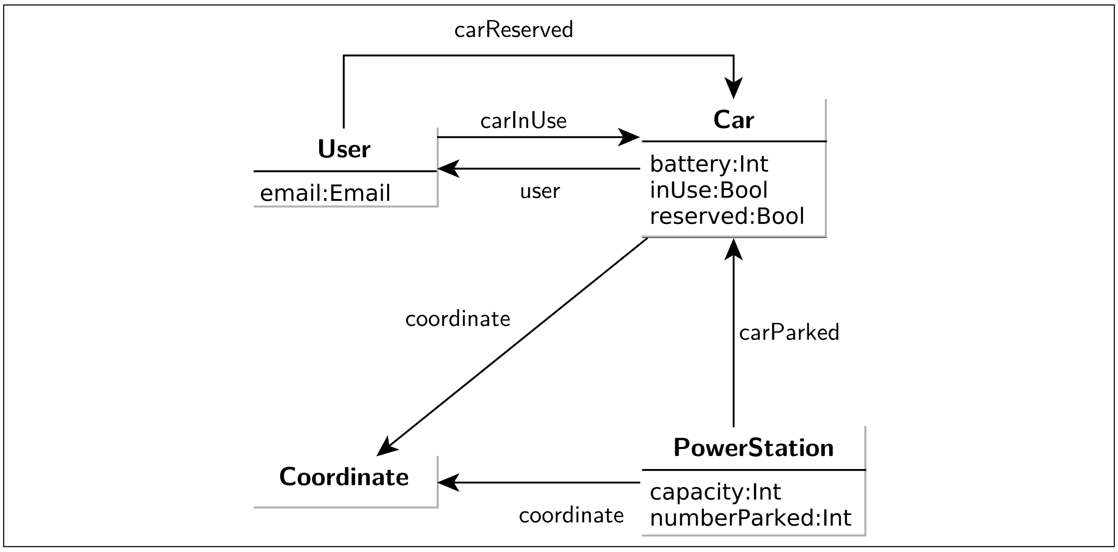
- Client applications should be user friendly, so that anyone can use them without a training.
- The system should be available 24/7, because customers may need any time to rent a car.
- The system should be scalable, because as the business grows, the number of requests to manage and the number of cars grows.
- All personal data should be stored in a secure way, according to the law.
- The system should be very protected from external attacks, in particular to prevent the theft of cars.

3.6. Alloy modelization and analysis

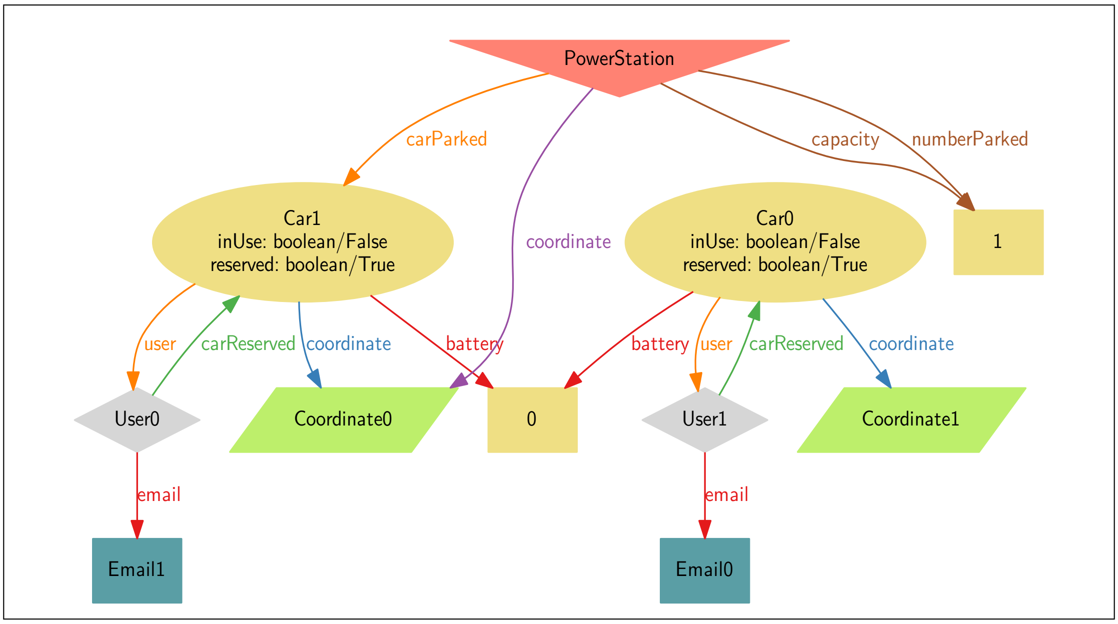
Part of the requirements have been formalized in the Alloy language. In this way it is simpler to design and implement a consistent product.
According to [Jack12] there are two types of models: static and dynamical ones. In particular static ones represent a snapshot of our world at a certain time while dynamical models represent the evolution of the system from one state to another one.

3.6.1. Static analysis

We have modelled in Alloy language this part of our world.



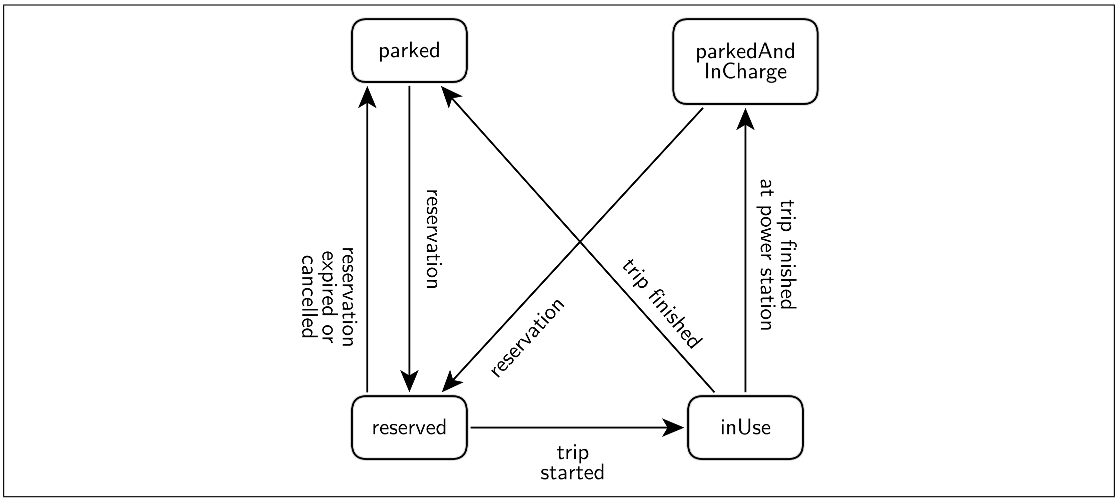
Below there is one of the possible different static models generated with `showStatic` *predicate.*



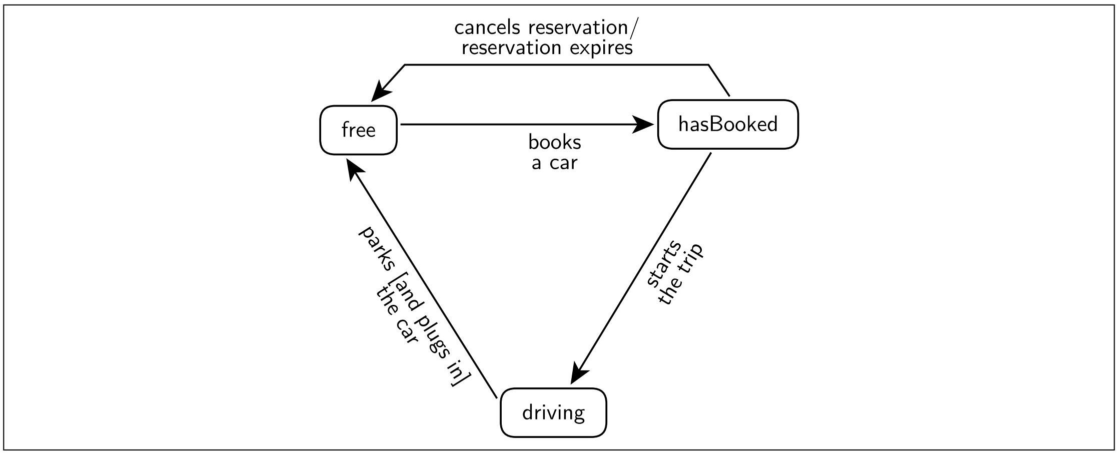
3.6.2. Dynamical analysis

Different actions have been analysed. They can be represented with these state diagrams.

- **Car**

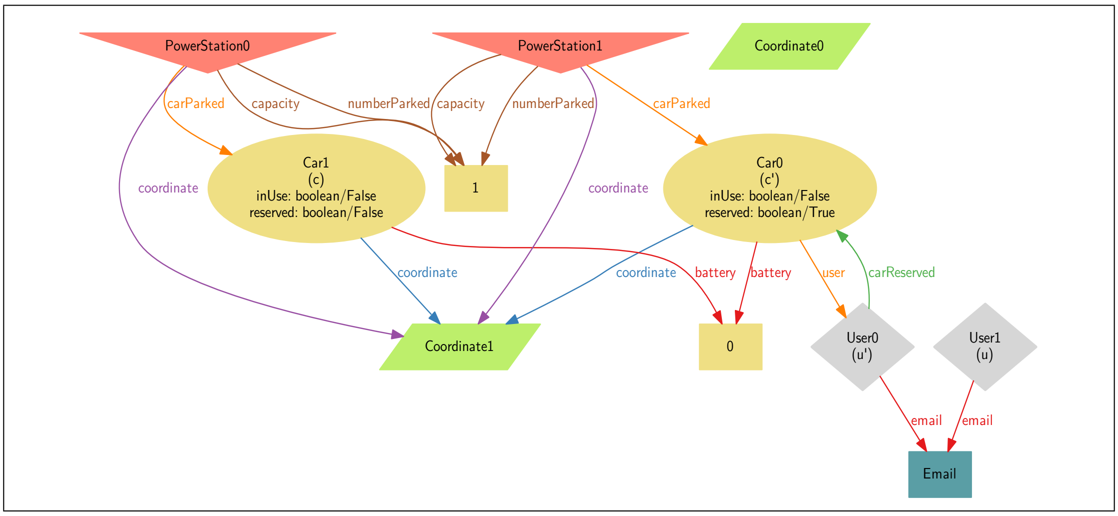


- **User**

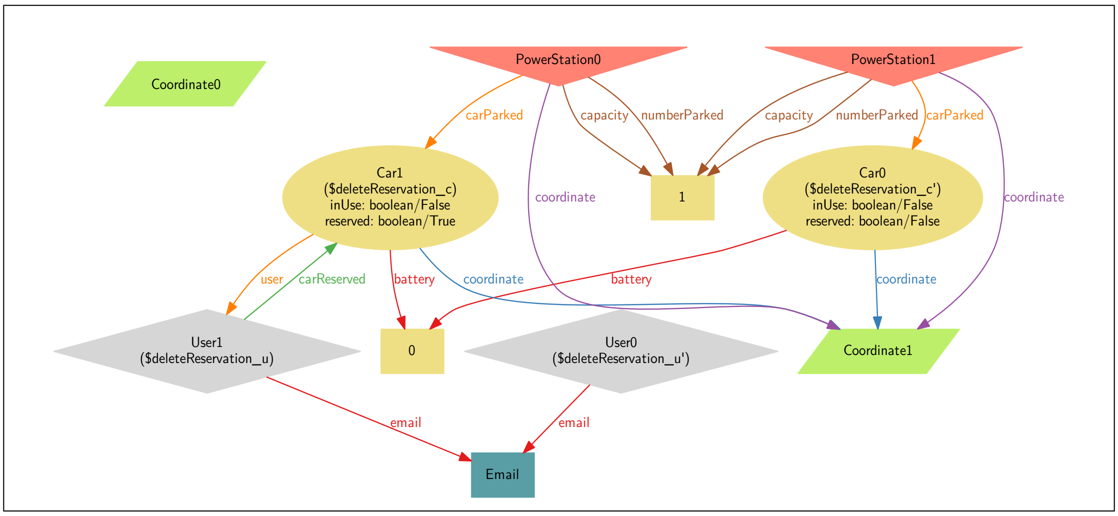


These are some examples of transitions as worlds generated by the Alloy solver. `c` and `u` (and all is attached to them) are always referred the pre-state while `c'` and `u'` (and all is attached to them) are referred to the post-state.

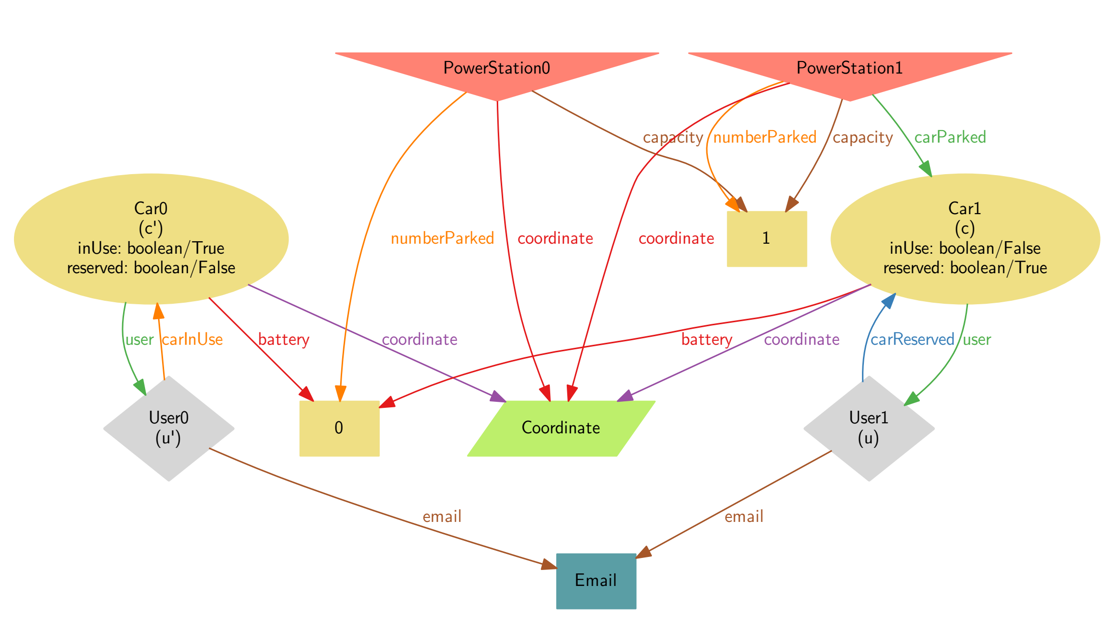
- **Reserve a car**



- **Delete a reservation**

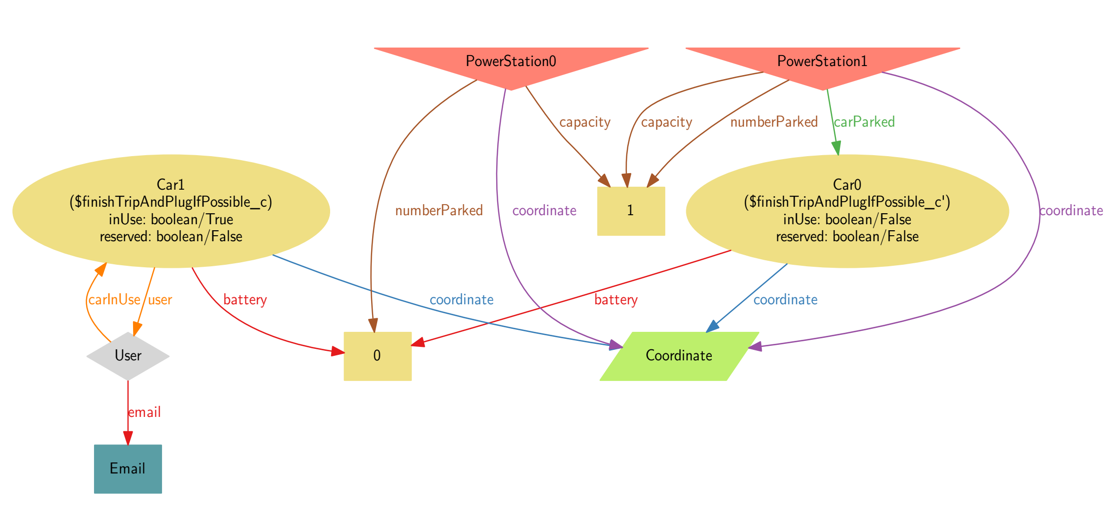


- **Start using a reserved car**

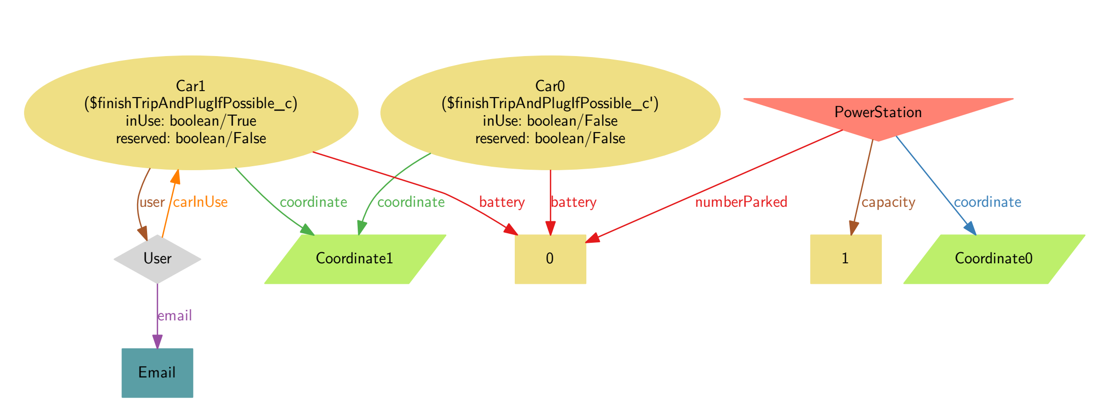


- **Finish the trip and plug the car to the station if there is one**

In this model there *is* the possibility to plug the car to the station.



In this model there *is not* the possibility to plug the car to the station.



3.6.3. Consistency results

All predicates are consistent and assertions have been verified with a scope constraint of 10. For more details on the implementation see Appendix A.

4. Effort spent

Component	Time spent (in hour)
Philippe Scorsolini	23
Lorenzo Semeria	18
Gabriele Vanoni	24.5

5. Changelog:

01/01/2017: Added "world and machine" analysis

A. Alloy source code

```
open util/boolean

sig Car {
	coordinate: one Coordinate,
	battery: one Int,
	inUse: one Bool,
	reserved: one Bool,
	user: lone User
} {
	battery >= 0
	battery =< 100
}

sig Coordinate {}

sig User {
	email: one Email,
	carInUse: lone Car,
	carReserved: lone Car
}

fact SimmetryBetweenDriverAndCar {
	all u:User | #u.carReserved=1 => u.carReserved.user=u
	all u:User | #u.carInUse=1 => u.carInUse.user=u
	all c:Car | #c.user=1 and c.inUse.isTrue => c.user.carInUse=c
	all c:Car | #c.user=1 and c.reserved.isTrue => c.user.carReserved=c
}

fact InUseAndReservedSemantics {
	all c:Car | (#c.user=1 => (c.inUse.isTrue or c.reserved.isTrue))
		and (c.reserved.isTrue <=> #c.user.carReserved=1)
		and (c.inUse.isTrue <=> #c.user.carInUse=1)
}

fact ReservationAndUseAreMutuallyExclusive {
	all c:Car | (c.reserved.isTrue => c.inUse.isFalse)
		and (c.inUse.isTrue => c.reserved.isFalse)

	all u:User | (#u.carInUse=1 => #u.carReserved=0)
		and (#u.carReserved=1 => #u.carInUse=0)
}

sig Email {}

pred oneUserPerEmail {
	all u1, u2: User | u1!=u2 => u1.email != u2.email
}
```

```
sig PowerStation {
       capacity: one Int,
       coordinate: one Coordinate,
       carParked: set Car,
       numberParked: one Int
} {
       numberParked=#carParked
       numberParked=<capacity
       capacity>0
       all c:Car | c in carParked => c.inUse.isFalse
}

pred eachPowerStationIsInADifferentPlace {
       all p1, p2: PowerStation | p1!=p2 =>
              p1.coordinate!=p2.coordinate
}

fact CarAtPowerStationHaveItsCoordinate {
       all c:Car, p:PowerStation | c in p.carParked =>
              c.coordinate=p.coordinate
}

fact CarCanBePluggedToOnlyOneStationAtOnce {
       all c:Car | lone p:PowerStation | c in p.carParked
}

pred showStatic {
       eachPowerStationIsInADifferentPlace
       oneUserPerEmail
}

pred reservedToInUse (c,c':Car, u,u':User) {
       c.reserved.isTrue
       c'.inUse.isTrue
       c.user=u
       c'.user=u'
       c.coordinate=c'.coordinate
       c.battery=c'.battery
       u.carReserved=c
       u'.carInUse=c'
       u.email=u'.email
       unplugCar[c]
}

pred unplugCar (c:Car) {
       all p:PowerStation | c in p.carParked => one p':PowerStation | (
              p'.carParked=p.carParked-c and
              p.capacity=p'.capacity and
```

```
                p.coordinate=p'.coordinate and
                p.numberParked=plus[p'.numberParked,1]
        )
}

pred finishTripAndPlugIfPossible (c,c':Car) {
        c.inUse.isTrue
        c'.inUse.isFalse
        c'.reserved.isFalse
        c.coordinate=c'.coordinate
        c.battery=c'.battery
        plugCar[c']
}

pred plugCar (c':Car) {
        all p,p':PowerStation | p.capacity=p'.capacity and
                p.coordinate=p'.coordinate

        all p:PowerStation | p.coordinate=c'.coordinate and
                c' not in p.carParked => one p':PowerStation | (
                p'.carParked=p.carParked+c' and
                p'.numberParked=plus[p.numberParked,1] )

        all p:PowerStation | c' in p.carParked => one p':PowerStation | (
                p'.carParked=p.carParked-c' and
                p'.numberParked=minus[p.numberParked,1] )
}

pred reserveCar (c,c':Car, u,u':User) {
        #c.user=0
        c.inUse.isFalse
        c.reserved.isFalse
        #u.carReserved=0
        #u.carInUse=0
        c'.user=u'
        c'.reserved.isTrue
        c'.inUse.isFalse
        c.coordinate=c'.coordinate
        c.battery=c'.battery
        u'.carReserved=c'
        #u.carInUse=0
        u.email=u'.email
        powerStationEvolution[c,c']
        powerStationEvolution[c',c]
}

pred powerStationEvolution (c,c':Car) {
        all p:PowerStation | (c in p.carParked => (one p':PowerStation | (
                p'.carParked = p.carParked - c + c' and
```

```
              p.capacity=p'.capacity and
              p.coordinate=p'.coordinate and
              p.numberParked=p'.numberParked )
       ))
}

pred deleteReservation (c,c':Car, u,u':User) {
       reserveCar[c',c,u',u]
}

assert BijectionBetweenUserAndEmail {
       showStatic => all u1,u2:User | u1=u2 <=> u1.email=u2.email
}

assert CarNotReservedAndInUse {
       all c:Car | not (c.reserved.isTrue and c.inUse.isTrue)
}

assert UserNotReservedAndInUse {
       all u:User | not (#u.carInUse=1 and #u.carReserved=1)
}

run showStatic for 2 Coordinate, 2 Car, 2 Email, 2 User,
       2 Int, 1 PowerStation

run reservedToInUse for 1 Coordinate, 2 User, 2 Car, 1 Email,
       2 PowerStation, 2 Int

run finishTripAndPlugIfPossible for 2 Coordinate, 1 User,
       2 Car, 1 Email, 2 PowerStation, 2 Int

run reserveCar for 2 Coordinate, 2 User, 2 Email, 2 Car,
       2 PowerStation, 2 Int

run deleteReservation for 2 Coordinate, 2 User, 2 Email, 2 Car,
       2 PowerStation, 2 Int

check BijectionBetweenUserAndEmail for 10

check CarNotReservedAndInUse for 10

check UserNotReservedAndInUse for 10
```