POLITECNICO
MILANO 1863

PowerEnJoy Project

Requirements Analysis and Specifications Document (RASD)

Alfredo Maria Fomitchenko (mat. 874656)

Version: 1.0

Release date: 13 November 2016

Description of the given problem

I am about to develop the PowerEnJoy project, regarding a service based on mobile applications that allows customers to rent electric cars in Milan and obtain ride discounts for virtuous behaviors.

The system provides registration and log-in forms, by which customers access the service and search for the location of available to be reserved cars.

After getting nearby a reserved car, users can unlock the car and perform a ride with it, keeping track of the charges based on the fixed minute rate by the screen inside the vehicle and at the end leaving the car in a safe area where recharging stations are available.

Users are incentivized to behave virtuously, receiving discounts if they either take with them at least two other passengers or leave the car with no more than 50% of the battery empty.

Definitions, acronyms, abbreviations

In this document the following terms will be used:

- "customer" = the person who registers to the system for benefiting from the provided service. He is the person who access to and drives the rented car taking therefore rights and duties specified in the scope of this document
- "driver" = often used as synonym for "Customer"
- "ride" = the customer's action he is charged for. It starts with the car engine ignition, and it ends with the door closing after parking the car in a safe area and leaving it under secure conditions (see [A17])
- "passengers" = one or more people allowed by the customer to enter the car and share the vehicle before the start of a ride
- "car" = the physical device rented to the customers for benefiting from the provided service. I picked 2015-2016 Kia Soul EV as the actual employed car model within the scope of this project (see "External Interface Requirements" paragraph).

- "car sensors" = set of car sensors integrated to the car Android Auto operating system capable of detecting the corresponding car part states (see "External Interface Requirements" paragraph).

  A sensor is defined to be either active or not; the meaning of a sensor activation is strictly related to security connotations, i.e. a car can be considered left after a ride under secure conditions if all its sensors are not active.

  This general definition readily leads to the following specific ones:

  - an engine sensor is active if the engine is on (for security concerns the engine must not be ignited);
  - a seat sensor is active if the seat is occupied by a person (for security concerns nobody must remain inside);
  - a door sensor is active if the door is opened (for security concerns all doors must be closed);
  - a window sensor is active if the window is opened (for security concerns all windows must be closed);
  - a parking brake sensor is active if the parking brake is not active (for security concerns the car must be left in a stationary condition).

  This mentioned scope of sensors has been reduced thanks to [A17] to the engine and the seat sensors set, "functional" to [RE14] and [RE22].

- “customer application” = the mobile application by which customers register to and access the system and benefit from the provided service
- “car application” = the Android-Auto-compatible application by which the system displays the current ride charge and gathers information about the car sensor states and the GPS location.
- “API” = acronym for Application Programming Interface, communication channel by which a system allows intercommunication with other systems.
- GPS position / GPS location / Location = referred to either customers and cars, location provided by the GPS hardware integrated inside either customers’ smartphones and cars’ standard equipment
- “safe area” = pre-defined set of locations which delimits a city area where a car location is marked as safe by the system
- “city” = administrative division of Milan

Assumptions, domain properties, constraints

General

[A1] Information provided by the customers are correct.

[A2] Requests and actions performed by the customers' accounts are interpreted as the customers' will.

[A3] Customers' and passengers' behaviors are fair towards the vehicles they use.

[A4] Driving license numbers provided by customers correspond to currently valid documents issued by the competent Italian authority.

[A5] The system embraces regulatory policies in accordance with the Data Protection Code (Legislative Decree no. 196/2003) and other related privacy laws.

[A6] The GPS position provided by any involved device is correct.

[A7] The GPS position associated to the customers' smartphones is equivalent to and interpreted as the customers' actual locations.

[A8] The system is able to communicate with Google Maps service and finalize requests such as managing GPS positions of customers and cars and converting a string of text into a location.

[A9] The system communicates with the Italian banks within the customer base's scope in order to request financial transactions based on the provided credit card numbers (see [A1]).

[A10] Financial assets associated to the credit card number provided by a customer are sufficient to let the system perform the financial transaction associated to a terminated ride.

[A11] The ride charge rate per minute is given and fixed.

[A12] The distance range within which a customer is willing to reserve an available car (see [G2]) is given and fixed for any reservation request.

Customers

[A13] Customers willing to start a ride make sure they have the necessary smartphone Internet connection capabilities and sufficient battery charge to correctly terminate the ride.

[A14] Customers willing to start a ride enter and let any passengers enter the car after unlocking it and before igniting the engine.

[A15] Customers do not leave in a safe area their in usage cars unless they are willing to terminate the ride.

[A16] Customers willing to terminate a ride exit the car after parking it inside allowed parking lines belonging to a safe area.

[A17] Customers willing to terminate a ride make sure they correctly leave the car under secure conditions, i.e. without active car sensors (see "Definitions, acronyms, abbreviations" paragraph).

[A18] Customers' behavior to the reservation expiration policy is fair, i.e. they tell the system they are near a previously reserved car when they are able to enter it in under one minute.

[A19] Customers' behavior to the passengers discount policy is fair, i.e. the number of passengers carried by the customer doesn't change throughout a ride.

[A20] Customers' behavior to the passengers discount policy is fair as they do not attempt to fool the car seat sensors into detecting passengers. In this respect, customers carry any package in the trunk, so that an active seat sensor (see "Definitions, acronyms, abbreviations" paragraph) implies the presence of a passenger.

Cars

[A21] The car operating system cannot be manually turned off.

[A22] The car operating system correctly works.

[A23] The car sensors cannot be manually turned off.

[A24] The car sensors correctly works.

[A25] The car operating system correctly detects a car sensor state.

[A26] The safe parking areas set is fixed and managed by the system.

[A27] Any ride is performed within the available car battery charge range.

[A28] After a ride, customers plug the car into the recharging station power grid and it is left recharging for at least the time necessary to perform the following ride.

[A29] Cars distribution throughout the city is fair and satisfies the customers needs.

Goals

[G1] Allow customers to register and receive a password by which log into the system.

[G2] Allow customers to retrieve a list of available cars and their positions given customers' positions (provided by GPS or as a specific address) within the distance range mentioned in [A12].

[G3] Allow customers to reserve a car from the list of available cars mentioned in [G2].

[G4] The system shall charge the customer 1 EUR and tag the reserved car available if he lets the reservation expire (i.e. after one hour).

[G5] Allow customers to unlock a previously reserved car (see [G3]) if the reservation has not expired yet (see [G4]).

[G6] Allow customers to keep track of the ride charges through the car screen.

[G7] Allow customers to stop being charged as soon as they park the car in a safe area and exit it.

[G8] Customers shall get a 10% discount if they bring at least two other passengers for the ride.

[G9] Customers shall get a 20% discount if they leave the car with no more than 50% of the battery empty.

Requirements

External Interface Requirements

To provide the described functionalities, the system relies upon two physical devices:

1) customers' smartphones;
2) rented cars.

The common denominator that allows full compatibility for the communication among these entities and the system is the Android platform: the customer mobile application, fundamental means for the customer to access the service, will be developed in Java to support Android smartphones; likewise rented cars will be equipped with Android Auto car system and the car application which will communicate with the system via the customer's smartphone Internet connection (see [A13]).

For this purpose, I chose the 2015-2016 Kia Soul EV as the most suitable device to be used within this project: the car received last September full Android Auto compatibility via software update and it represents a cost-effective starting-point car on the business side.

Therefore, relying on this platform makes on-the-go customers' requests and gathering all the information needed by the system to determine car locations and sensors state natural and straightforward.

Also, this leads to a readily communication integration towards the two involved external actors:

1) Google Maps, whose API are fundamentally intertwined with Android and the Google ecosystem in general;
2) customers' credit card banks, which all provide compatibility with online financial transaction requests.

Functional requirements

The following functional requirements can be derived to fulfill [G1] through [G9] assuming that [A1] through [A29] hold:

[G1] Allow customers to register and receive a password by which log into the system.

- [RE1] The system shall let a new customer register.
- [RE2] The system shall generate, store and send to the new registered customer his associated password.
- [RE3] The system shall let a customer log in if he correctly provides the email address and the associated password.

[G2] Allow customers to retrieve a list of available cars and their positions given customers' positions (provided by GPS or as a specific address) within the distance range mentioned in [A12].

- [RE4] The system shall request the appropriate customer application permissions to obtain and manage the customers' positions.
- [RE5] The system shall locate a customer by the GPS position.
- [RE6] The system shall convert a specified address provided by a customer into a specific location through Google Maps service (see [A8]).
- [RE7] The system shall retrieve and provide to a customer the list of available car from the customer's position within a fixed distance range.

[G3] Allow customers to reserve a car from the list of available cars mentioned in [G2].

[RE8] The system shall verify if a customer neither has previously reserved nor is currently using a car.

[RE9] The system shall process a customer's request of reservation associating the car to the customer, tagging it as unavailable and starting an associated reservation timer.

[G4] The system shall charge the customer 1 EUR and tag the reserved car available if he lets the reservation expire (i.e. after one hour).

[RE10] The system shall stop the reservation timer associated to the reservation (see [RE9]) after one hour if the system has not stopped it yet (see [RE13]).

[RE11] After stopping the reservation timer as in [RE10], the system shall charge the customer 1 EUR.

[RE12] After stopping the reservation timer as in [RE10], the system shall tag the car associated to the reservation as available.

[G5] Allow customers to unlock a previously reserved car (see [G3]) if the reservation has not expired yet (see [G4]).

[RE13] The system shall unlock the car when the customer communicates that he is close enough (see [A18]) and therefore stop the reservation timer (see [RE9]).

[G6] Allow customers to keep track of the ride charges through the car screen.

[RE14] The system shall start a ride charge timer as soon as the car operating system communicates to the system that the engine has been ignited by the customer.

[RE15] The system shall compute the current ride charge by means of the ride charge timer (see [RE14]) and the given charge rate per minute (see [A11]).

[RE16] The system shall display the current ride charge through the car application.

[G7] Allow customers to stop being charged as soon as they park the car in a safe area and exit it.

[RE17] The system shall locate a car via its GPS position.

[RE18] The system shall determine if a car location is within a safe area or not.

[RE19] The system shall lock the car after detecting through the car application that the car engine has been turned off inside a safe area (see [A16], [A17]).

[RE20] The system shall stop the ride charge timer when the car has been locked by the system (see [RE14], [RE19]).

[RE21] After stopping the ride charge timer as in [RE20], the system shall request the charge as a financial transaction to the customer's credit card bank after verifying any applicable discount (see [G8], [G9]).

[G8] Customers shall get a 10% discount if they bring at least two other passengers for the ride.

[RE22] The system shall detect the car seat sensors states through the car application at the beginning of the ride and there determine the number of passengers of the ride (see [A19], [A20]).

[RE23] The system shall apply the mentioned discount to the ride charge before the charge has been requested to the credit card bank (see [RE21]).

[G9] Customers shall get a 20% discount if they leave the car with no more than 50% of the battery empty.

[RE24] The system shall detect through the car application the car battery charge.

[RE25] The system shall apply the mentioned discount to the ride charge before the charge has been requested to the credit card bank (see [RE21]).

Non functional requirements

[NFR1] The system shall be available 24 hours a day, 7 days a week.

[NFR2] The system shall send the associated passwords to the newly registered customers' email addresses in under 5 minutes.

[NFR3] The system shall support parallel operations from different customers at the same time, with response delays of at most 10 seconds.

[NFR4] The system Requirements Analysis and Specification Document and Design Document shall be drafted to understand the given problem, focus on the particular requirements and specifications defined by the goals and define the real structure of the system and its layers.

Scenarios

Here are some possible applications of usage of the service provided by the described system.

Scenario 1

John is feeling weird while waking up in his bed, as the sun has risen more than the previous morning. In fact, it has since his smartphone alarm clock decided not to go off for some inexplicable reason.
He gets out his home in a rush predicting he is not going to make it to work as the metro is always a jam at that hour of the day. However, after registering to the PowerEnJoy service the day before to use it sooner or later, the application happens to be on his smartphone and John couldn't be better off than trying it right now.
He taps on the "Reserve it!" button associated to the car he can see right across the street, and by the time he approaches it the "Unlock it!" button works flawlessly letting him enter without making his rush stop.
The ride will be long, but not as long as the metro's.

Scenario 2

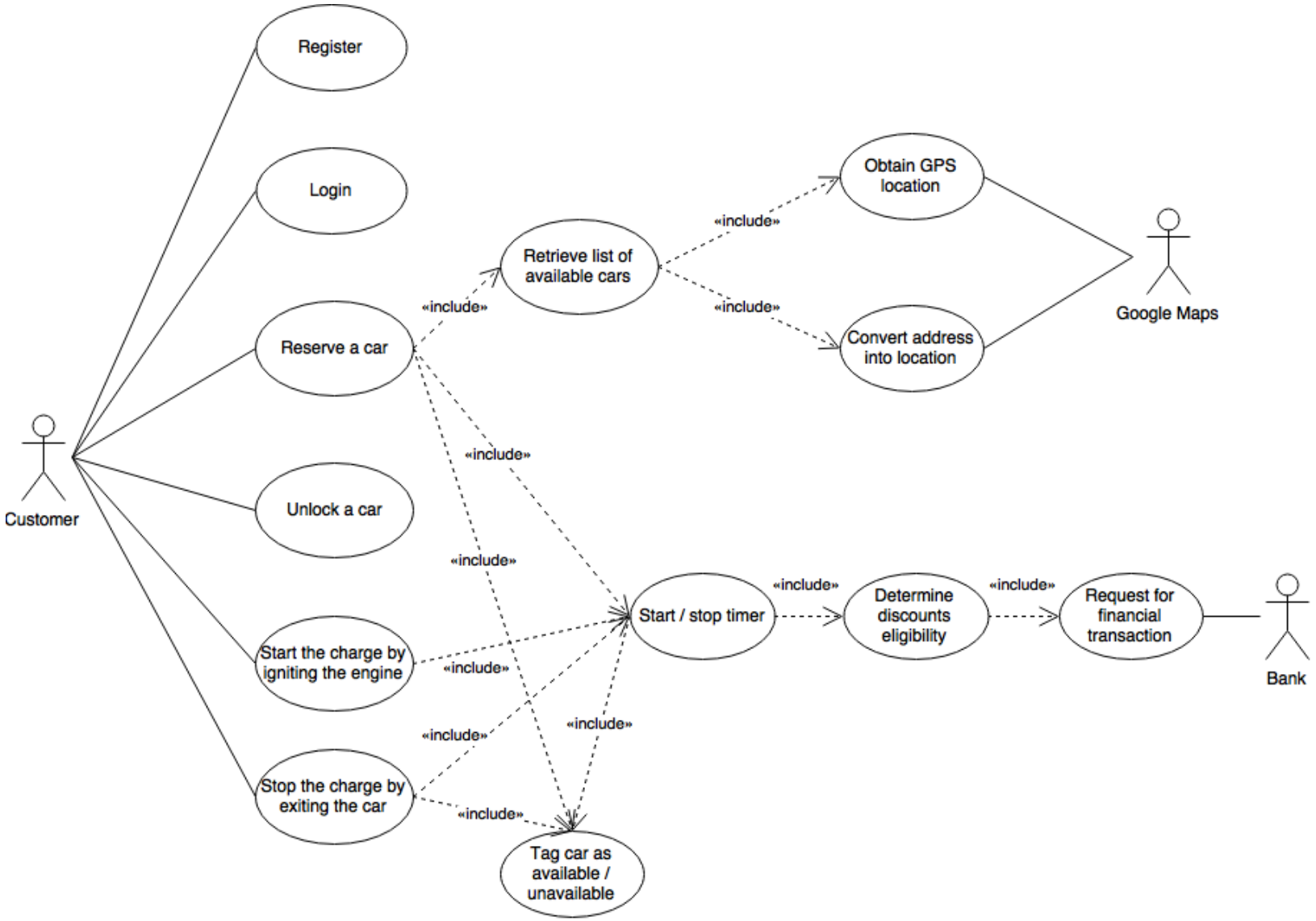
Plans for lunch must be different today: John's manager is in town. He and his colleagues know he cannot help loving Mario's place pasta right next to Duomo but how are they supposed to find parking in that area?
John knows the answer: PowerEnJoy cars are allowed to park even inside the parking lines restricted to residents only. So John reserves a car and tells his colleagues to do so, as the 12:00 PM manager's meeting is due in thirty minutes.

Scenario 3

The night out is over but the last metro train is not going to wait for John since he yet has to say goodbye to his manager and colleagues before trying to run and catch it.
The Friday night crowd does not leave many PowerEnJoy cars available in that area and fairly enough the application retrieve none as John gives his current position to search for cars with. After some manual location research due to misspellings (he is still able to drive home safely though) he manages to reserve a not so near car, at least giving him the opportunity to take a night walk in that beautiful area of the city.

UML models

Use cases diagram



Use cases descriptions

Following are the descriptions of some use cases represented in the above diagram:

The customer registers for an account

Name: The customer registers for an account

Actors: Customer

Entry conditions: There are no entry conditions.

Flow of events:

1) The customer opens the previously installed customer application on his smartphone and reaches the homeActivity
2) The customer taps on the "Register" button
3) The customer fills in the form with all the required information that the customer application is asking for.
4) The customer taps on the "Register me!" button

Exit conditions: A "Your registration has been submitted. Please check your email inbox for further information" message is successfully shown to the customer.

Exceptions:

a) The customer has not entered consistent data inside the form (e.g. an email address that does not show the "@" character) or he has entered incomplete information. An error message is shown to the customer warning him to double-check the provided information.

The customer logs into the system

Name: The customer logs into the system

Actors: Customer

Entry conditions: There are no entry conditions.

Flow of events:

1) The customer opens the previously installed customer application on his smartphone and reaches the homeActivity
2) The customer taps on the "Login" button
3) The customer enters his email address and the associated password inside the form the customer application is showing
4) The customer taps on the "Log me in!" button

Exit conditions: The customer successfully logs into his own private page inside the customer application.

Exceptions:

a) The customer has not entered the correct either email address or password. An error message is shown to the customer warning him to double-check the provided information.

The customer reserves a car

Name: The customer retrieves the list of available cars

Actors: Customer, Google Maps

Entry conditions: The customer logs into the system (see homonymous use case) and taps on the "Reserve a car" button.

Flow of events:

1) "Use current position" and "Insert address" buttons are shown to the customer.
2) If the customer taps on the "Use current position" button, he is prompted with a message asking for the permissions to access his current GPS location if not asked before. If the customer agrees, 4) occurs; if he does not, exception a) occurs.
3) If the customer taps on the "Insert address" button, he enters the address corresponding to the location he wants to search for cars from. If Google Maps cannot associate the string with a valid location, exception occurs
4) A map representing the locations of the available cars is shown to the customer.
5) The customer taps on an available car.
6) The customer taps on the "Reserve it!" button.
7) The system checks if the customer neither has reserved nor is using another car.

Exit conditions: The customer successfully reserves the desired car.

Exceptions:

a) The customer does not grant permissions to obtain the current GPS location associated to his smartphone. The customer is required by a warning message to close and open the customer application again and grant permissions.

b) The customer either has reserved or is using another car. An error message is shown to the customer suggesting to him to either pick the other car up or to terminate the current ride.

c) The customer has not manually entered a valid address. An error message is shown to the customer warning him to double-check the provided information and enter the address again.

The customer unlocks a previously reserved car

Name: The customer unlocks a previously reserved car

Actors: Customer

Entry conditions: The customer has reserved a car (see "The customer reserves a car" use case).

Flow of events:

1) The customer logs into the system and taps on the "Reservation" button.
2) The customer taps on the "Unlock it!" button associated to the previously reserved car.

Exit conditions: The customer successfully unlocks the car.

Exceptions: There are no exceptions.

The customer starts a ride

Name: The customer starts a ride

Actors: Customer

Entry conditions: The customer has unlocked a car (see "The customer unlocks a previously reserved car" use case).

Flow of events:

1) The customer and any passengers enter the previously unlocked car.
2) The customer ignites the engine by pressing the appropriate button inside the car.
3) The car operating system shows on the display next to the dashboard the current ride charge.

Exit conditions: The customer successfully start driving to the desired safe parking area.

Exceptions: There are no exceptions.

The customer terminates a ride

Name: The customer terminates a ride

Actors: Customer, Bank

Entry conditions: The customer approaches a safe parking area where he is willing to terminate the ride.

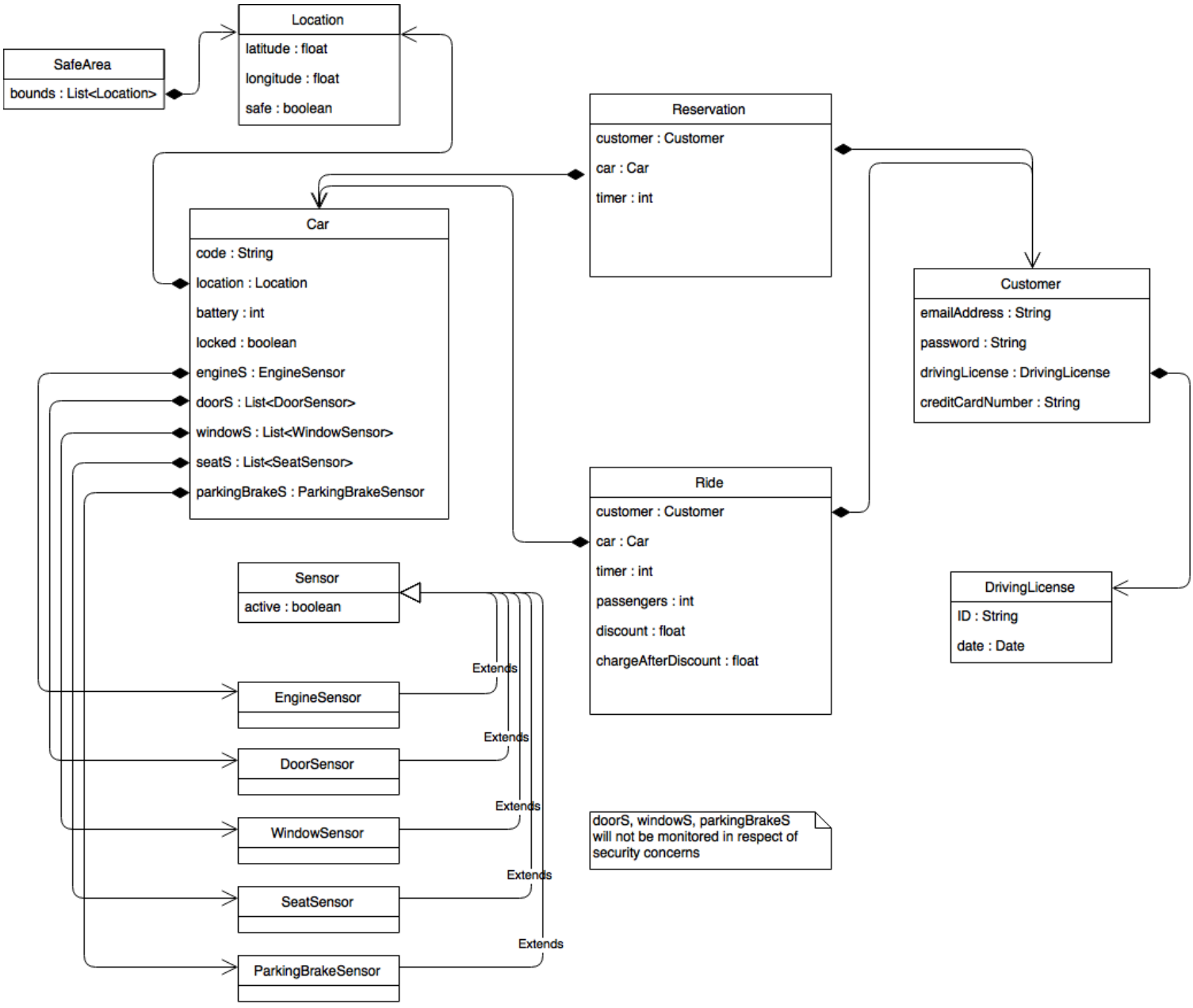
Flow of events:

1) The customer parks the car inside permitted parking lines within the safe area.
2) The customer exits and lets any passengers exit the car.
3) The customer closes the car driver door.
4) The system applies any discount to the current ride charge
5) The system requests the customer's bank for a financial transaction through his credit card number.

Exit conditions: The system performs the car lock and charges the customer for the ride.

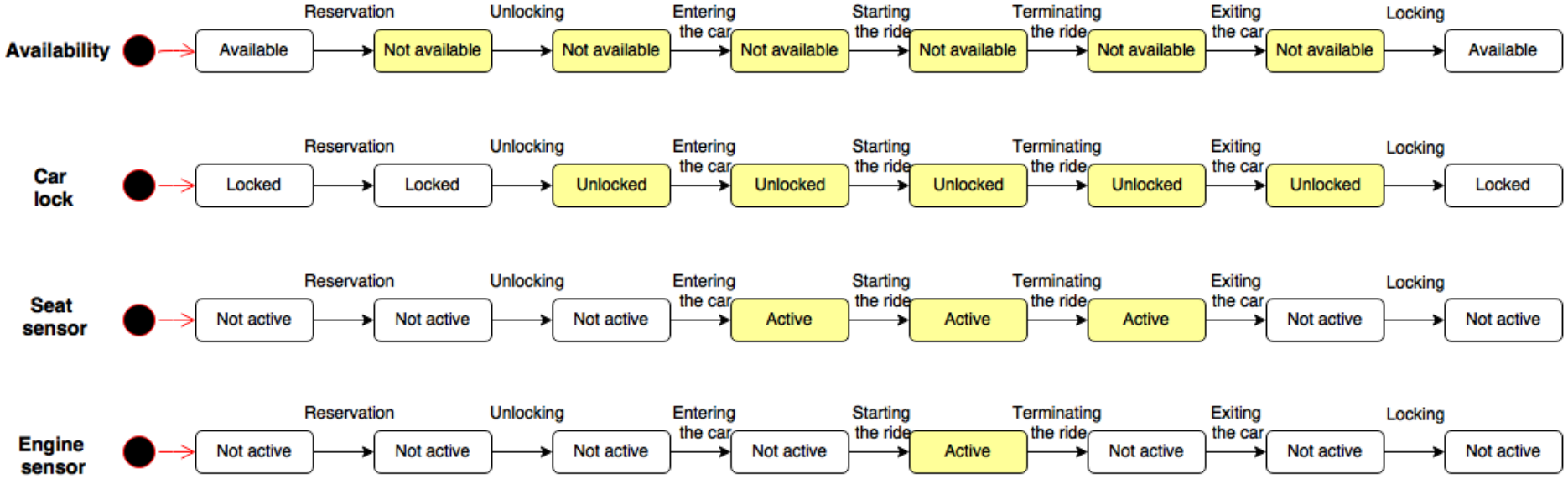
Exceptions: There are no exceptions (see [A16], [A17]).

Class diagram

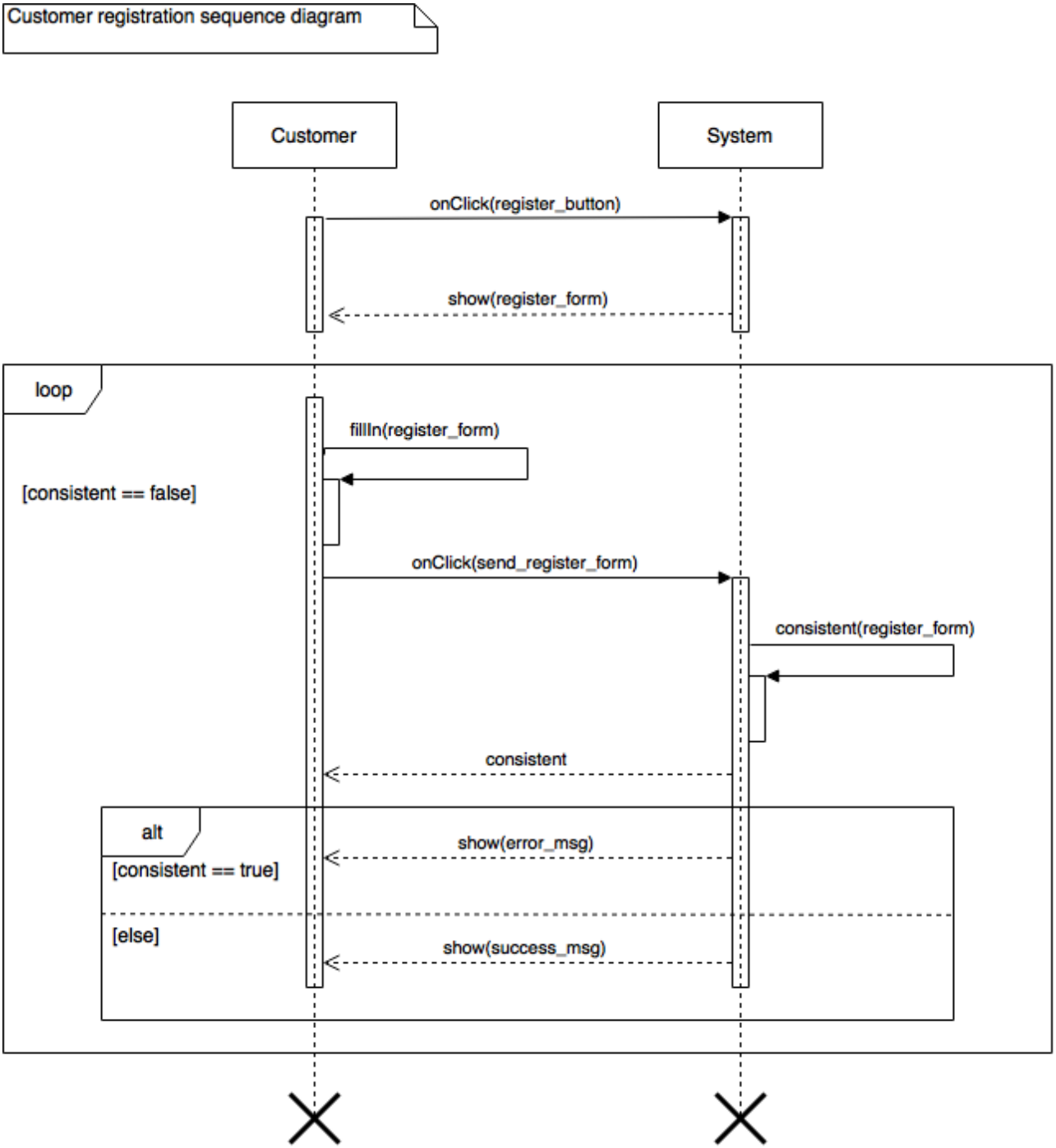


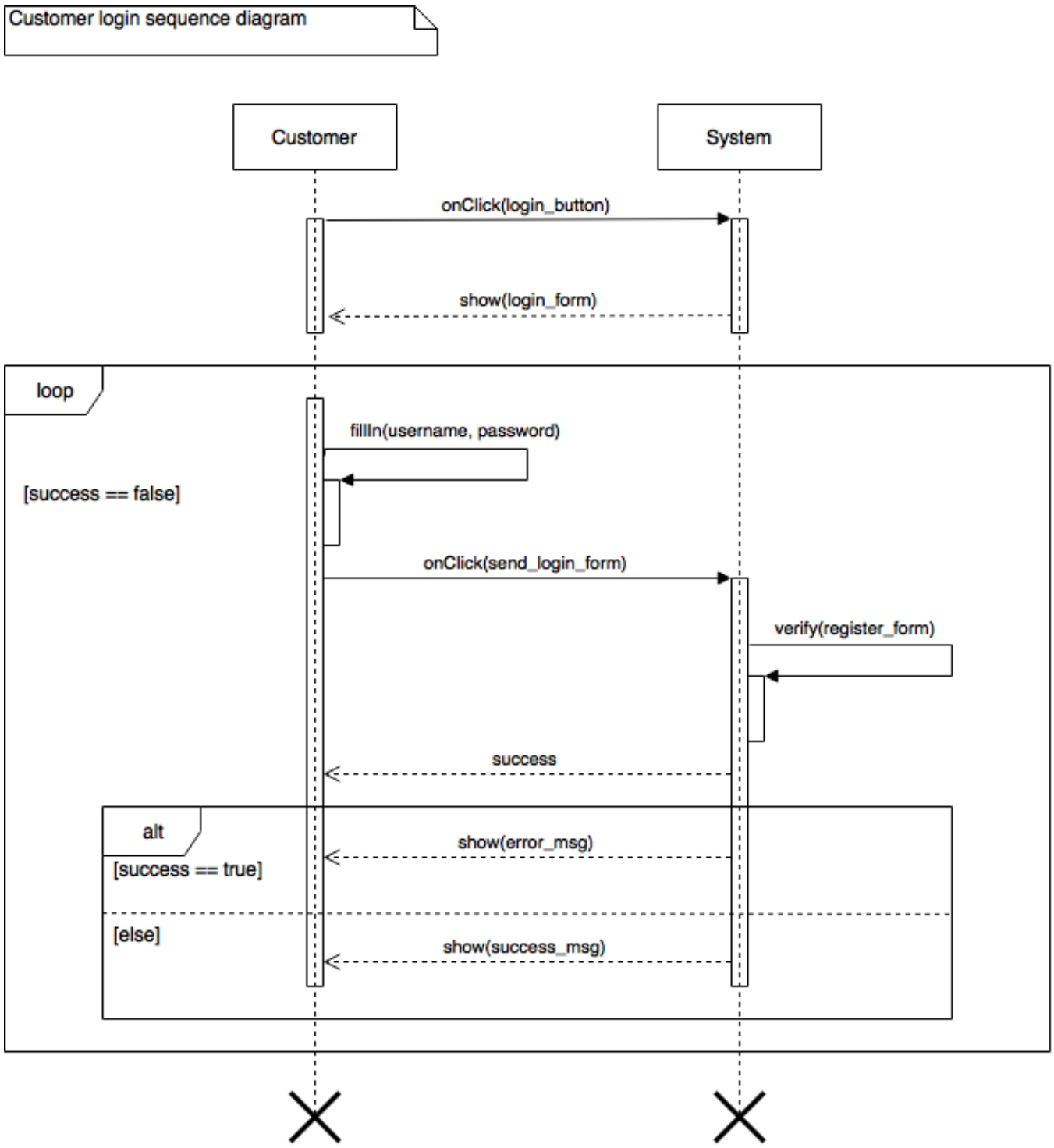
Statechart

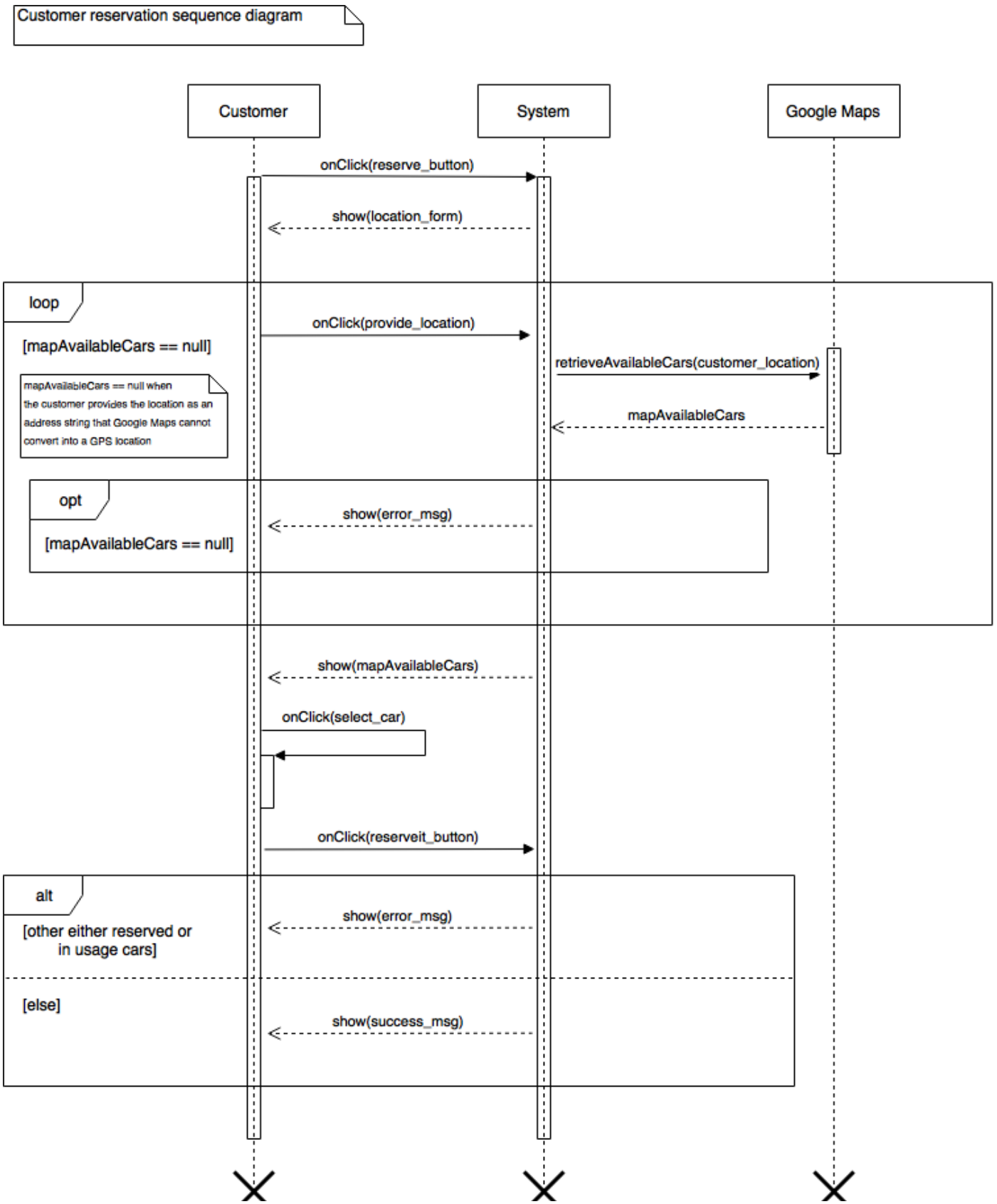
Cycling statechart of a car through reservation and usage

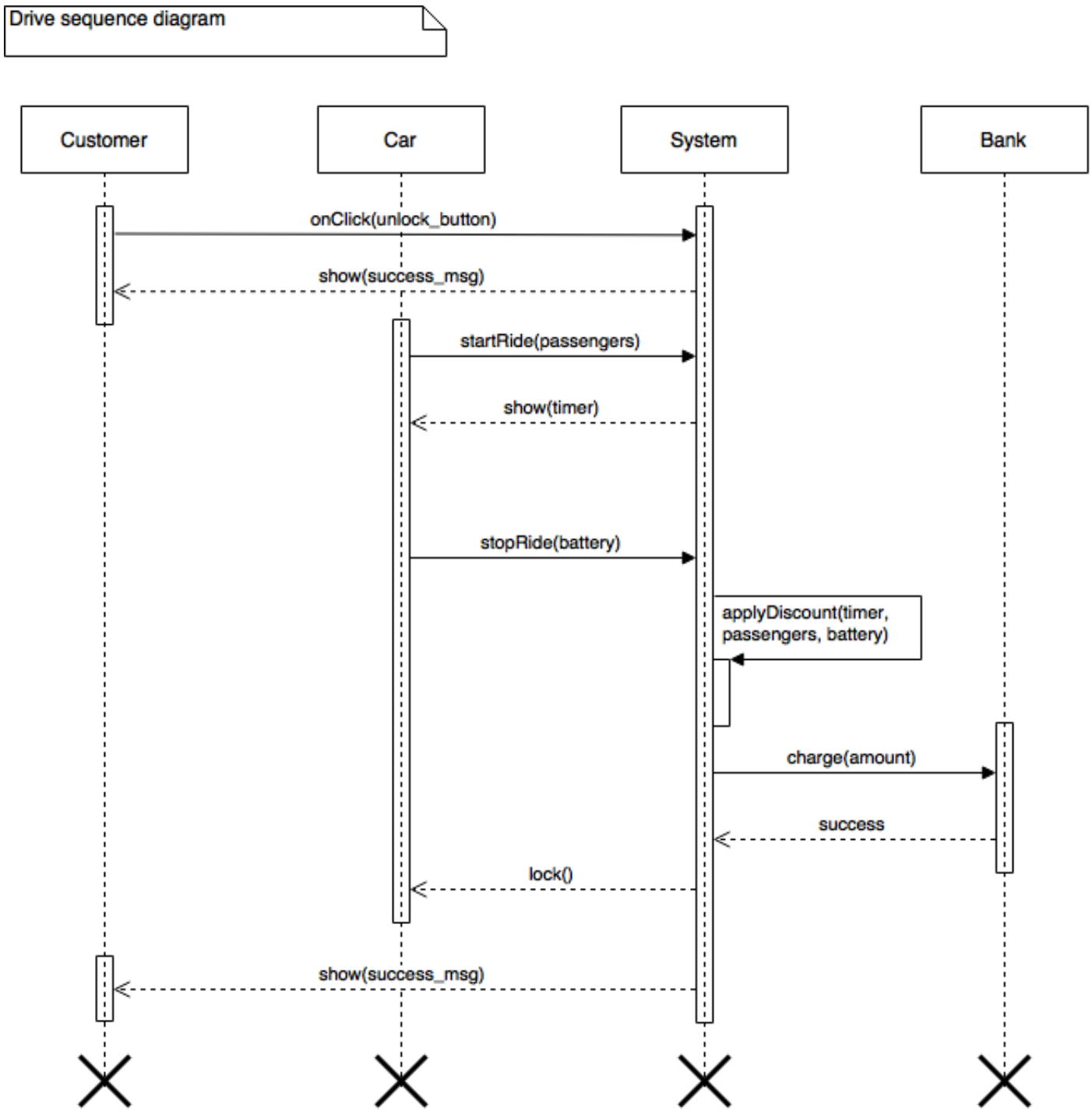


Sequence diagrams




Customer login sequence diagram
Customer
System
onClick(login_button)
show(login_form)
loop
fillIn(username, password)
[success == false]
onClick(send_login_form)
verify(register_form)
success
alt
show(error_msg)
[success == true]
[else]
show(success_msg)


Customer reservation sequence diagram
Customer
System
Google Maps
onClick(reserve_button)
show(location_form)
loop
[mapAvailableCars == null]
mapAvailableCars == null when the customer provides the location as an address string that Google Maps cannot convert into a GPS location
onClick(provide_location)
retrieveAvailableCars(customer_location)
mapAvailableCars
opt
[mapAvailableCars == null]
show(error_msg)
show(mapAvailableCars)
onClick(select_car)
onClick(reserveit_button)
alt
[other either reserved or in usage cars]
show(error_msg)
[else]
show(success_msg)


Drive sequence diagram
Customer
Car
System
Bank
onClick(unlock_button)
show(success_msg)
startRide(passengers)
show(timer)
stopRide(battery)
applyDiscount(timer, passengers, battery)
charge(amount)
success
lock()
show(success_msg)

Alloy model

```
open util/boolean

sig Location {
        latitude : Int, // should be float
        longitude : Int, // should be float
        safe : one Bool
}

sig Sensor { active : one Bool }
sig Car {
        available : one Bool,
        locked : one Bool,          // automatic door lock performed by the system
                                    // not the car doors state
        location : one Location,
        engineSensor : one Sensor,
        seatSensor : some Sensor
} {
// #seatSensor >= 4 would have increased the complexity of the diagram without any
gain in terms of information
}
sig Customer {
        location : one Location
}

sig Reservation {
        car : one Car,
        customer : one Customer
}
sig Ride {
        car : one Car,
        customer : one Customer
}
```

```
// Duplicates
// A car can only be reserved by one customer at a time, and a customer cannot reserve more than one car
fact noDoubleReservations { all r1, r2 : Reservation | r1 != r2 implies r1.customer != r2.customer and r1.car != r2.car }
// A customer cannot use more than one car at a time, and a car cannot be used by more than one customer at a time
fact noDoubleUsages { all r1, r2 : Ride | r1 != r2 implies r1.customer != r2.customer and r1.car != r2.car }
// A customer cannot use a car and reserve another one at the same time, and a car cannot be reserved while another customer is driving it
fact noDoubleReservationAndUsage { no reserv : Reservation, ride : Ride | reserv.customer = ride.customer or reserv.car = ride.car }

// Locations
// Locations are considered without approximations within the scope of this model
fact locationsAreSoAccurateTheyCannotBeEqual {
        all c1, c2 : Customer | c1 != c2 implies c1.location != c2.location
        all c1, c2 : Car | c1 != c2 implies c1.location != c2.location
}
// A customer's location is always different from a car location except for the case in which he is using the car
fact driversAreActuallyInsideTheCar { no cust : Customer, macch : Car | cust.location = macch.location and no r : Ride | r.customer = cust and r.car = macch }

// Availability
fact {
        // A car is available if neither reserved nor in use
        all c : Car | c not in Reservation.car and c not in Ride.car implies c.available.isTrue
        // A car is not available if reserved
        all c : Car | c in Reservation.car implies not c.available.isTrue
        // A car is not available if in use
        all c : Car | c in Ride.car implies not c.available.isTrue
        // As previously mentioned, the car lock is considered to be
        // the automatic one performed by the system
        // when the car is left under secure conditions, not the car doors state
        all c : Car | c in Ride.car implies not c.locked.isTrue
}

// Usage
// A car in use has at least a person inside
fact carsDontDriveThemselves { all c : Car | c in Ride.car implies some sens : c.seatSensor | sens.active.isTrue }
```

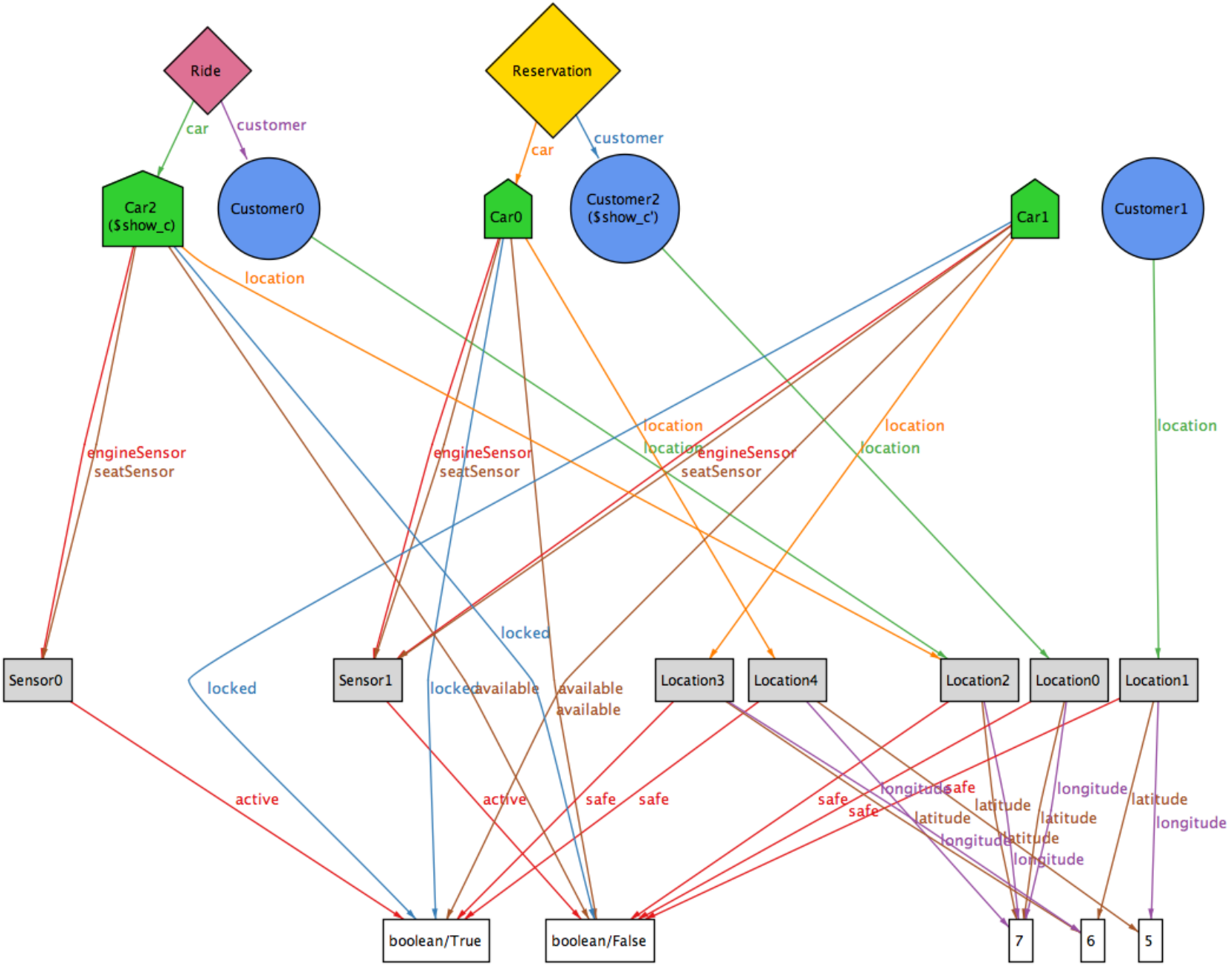
```
// Safety
// After a ride, a car is left under secure conditions (see "Definitions, acronyms, abbreviations")
fact carsAreLeftUnderSecureConditions {
        all c : Car | c not in Ride.car implies
                // The location is safe
                c.location.safe.isTrue and
                // Car doors are closed to let the system perform an automatic lock
                c.locked.isTrue and
                // The engine is not ignited
                one sens : c.engineSensor | not sens.active.isTrue and
                // Nobody is left inside
                all sens : c.seatSensor | not sens.active.isTrue
}


pred show() {
        #Reservation = 1
        #Ride = 1
        #Car = 3
        #Customer = 3
        some c : Car | not c.location.safe.isTrue
        some c : Customer | not c.location.safe.isTrue
}

run show for 5
```

```
Executing "Run show for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  6095 vars. 440 primary vars. 11164 clauses. 59ms.
  Instance found. Predicate is consistent. 37ms.
```

Generated world



Used tools, reference documents and resources

I used for this RASD as reference for the general layout and structure

- RASD sample from Oct. 20 lecture.pdf,
- Requirement Engineering Part III.pdf;

to create the UML diagrams

- draw.io website;

to prove the consistency and create the diagram of the Alloy model

- Alloy Analyzer 4.2;

to write the actual document

- Microsoft Word 2016;

as scheduling and time effort management tracker

- Microsoft Excel 2016.

Effort

-	16 October 2016:	**1 h**
-	17 October 2016:	**1 h**
-	20 October 2016:	**2 h**
-	23 October 2016:	**2,7 h**
-	24 October 2016:	**1,4 h**
-	26 October 2016:	**2,0 h**
-	27 October 2016:	**2,5 h**
-	28 October 2016:	**4 h**
-	29 October 2016:	**2 h**
-	30 October 2016:	**2,5 h**
-	2 November 2016:	**3 h**
-	3 November 2016:	**3,5 h**
-	4 November 2016:	**4 h**
-	5 November 2016:	**1,3 h**
-	6 November 2016:	**2,9 h**
-	7 November 2016:	**3,4 h**
-	8 November 2016:	**3,9 h**
-	9 November 2016:	**3,2 h**
-	10 November 2016:	**3,3 h**
-	11 November 2016:	**3,5 h**
-	12 November 2016:	**3,8 h**
-	13 November 2016:	**2 h**
		58,9 h

Changelog

- v1.0
- v1.0.1: added release date on the front page