

Version: 1.0.0  
Release date: 25-10-2016

Riccardo Cattaneo 873647

Fabio Chiusano 874294

PowerEnjoy

Summary

[Introduction 2](#_Toc465186928)

[Description of the given problem 2](#_Toc465186929)

[Goals 2](#_Toc465186930)

[Domain properties 3](#_Toc465186931)

[Glossary 3](#_Toc465186932)

[Text assumptions 0](#_Toc465186933)

[Constraints 0](#_Toc465186934)

[Regulatory policies 0](#_Toc465186935)

[Hardware limitations 0](#_Toc465186936)

[Interfaces to other applications 0](#_Toc465186937)

[Parallel operation 0](#_Toc465186938)

[Proposed system 0](#_Toc465186939)

[Identifying stakeholders 0](#_Toc465186940)

[Reference documents 0](#_Toc465186941)

[Actors identifying 0](#_Toc465186942)

[Requirements 0](#_Toc465186943)

[Functional requirements 0](#_Toc465186944)

[Non-functional requirements 0](#_Toc465186945)

[Scenario identifying 0](#_Toc465186946)

[UML models 0](#_Toc465186947)

[Use case diagram 0](#_Toc465186948)

[Use case description 0](#_Toc465186949)

[Class diagram 0](#_Toc465186950)

[Sequence diagrams 0](#_Toc465186951)

[Activity diagrams 0](#_Toc465186952)

[State diagrams 0](#_Toc465186953)

[Alloy modeling 0](#_Toc465186954)

[Model 0](#_Toc465186955)

[Alloy result 1](#_Toc465186956)

[World generated 1](#_Toc465186957)

[Future development 1](#_Toc465186958)

[Used tools 1](#_Toc465186959)

[Hours of work 1](#_Toc465186960)

[Changelog 1](#_Toc465186961)

# Introduction

## Description of the given problem

We will project the digital management system for PowerEnJoy™, which is a car-sharing service that exclusively employs electric cars.

The system, first, has to provide normally provided by car-sharing services such as the possibility for a new user to register and log in, to find locations of nearby available cars and to reserve them.

It has also to guarantee that a user who has used the service pays a fee that should be as fair as possible.

In order to strengthen the ecological mission of PowerEnJoy™, the system aims to incentivize virtuous behaviours of the users by adapting the final bill for every ride.

For example, if there are at least three people on the car, or if the car is left charging at special parking areas, the system has to apply a discount. Instead, if the car is left far from a charging station with a low battery level, it has to apply a charge on the bill.

## Goals

* Users could register to the system and have their personal area;
* Users could see and select an available car close to him, or close to a specified address, and reserve it for up to one hour before they pick it up;
* Users could get in a car only if they are beside it and they reserved it;
* Users should pay proportionally to minutes they have used the car, and they should see in real time the amount of the bill;
* Virtuous behaviours should be incentivized for users by paying, on the last ride:
  + 10% less if they share their trip with at least other two passengers;
  + 20% less if the car is left with at least 50% of battery level;
  + 30% less if the car is left plugged in at special parking areas;
  + 30% more if the car is left at more than 3km from the nearest power grid station with less than 30% of battery level.

## 

## Domain properties

We suppose that these conditions hold in the analysed world:

* All the users have a device connected to the Internet with a GPS built in;
* All the electric cars have a GPS to indicate their actual position, that cannot be turned off, and a sensor for every seat which detect the presence of a passenger;
* GPS position is always accurate;
* A car can be in only one zone at the same time and this is the real zone;
* Every car, when is first picked up, has the battery level at 100%;
* A car can’t be used by multiple users simultaneously;
* In a special parking area with power grid stations there’s always space for a car to be plugged in.
* Users behave politely and have no intention of cheating;
* The payment infos provided by the users are always valid;

## Glossary

* Management System: the management system of the car share service;
* User: a user is a person already registered in the system, so that has a profile, and sometimes is interested to reserve and use a car;
* Car: for “car”, “electric vehicle” or “electric car” we mean an electric car involved in PowerEnJoy™, ready to be reserved and used;
* Ride: usage of a car, by one user, that has to pay it with a bill. It starts when the user picks up a car after a resevation is made and it ends when the user leaves the car in a safe area;
* Reservation: is the ability of a user to reserve a car one hour prior to the pick up;
* Bill: compensation to be paid for a ride by the user;
* Guest: a guest is a person that probably for the first time accesses the system or that has not already signed up;
* Safe Area: area where a user can leave the car he’s renting;
* Special Safe Area or Safe Area with power grid station: Safe Area where the user can plug the car into the power grid station in order to get a discount on the ride;
* Battery level: how much in percentage the battery is charged;
* Passengers: people that are in a car during a ride. The user that drives the car is included in the passengers count.

## Text assumptions

* There already is a portal (web site, mobile app, …) that a guest can use in order to register to the system;

## Constraints

### Regulatory policies

The Management System must ask the users the permission to get their position and to manage sensible data (position).

### Hardware limitations

3G and GPS connections are required, then the system must be usable on top of a platform built for mobile systems.

### Interfaces to other applications (system boundaries)

The system relies on an external payment handler.

### Parallel operation

The server supports parallel operations from different clients

## Proposed system

## Identifying stakeholders

There are a lot of entities that incentivize the use of electric vehicles, since they are less harmful to the environment:

* The government.
* The city in which the service is active.

## Reference documents

# 

# Actors identifying

# 

# Requirements

## Functional requirements

## Non-functional requirements

# 

# Scenario identifying

# 

# UML models

## Use case diagram

## Use case description

## Class diagram

## Sequence diagrams

## Activity diagrams

## State diagrams

# Alloy modeling

## Model

Open util/boolean

sig Company {

cars: set Car,

safe\_areas: set Safe\_Area

}

sig Car {

id: Int,

available: Bool,

location: Location,

unlocked: Bool,

engine\_on: Bool,

battery\_level: Percentage

}

sig Percentage {

level: Int

} {

level >= 0,

level <= 100

}

sig User {

credential: Credential,

password: Password,

payment\_info: Payment\_Info,

location: Location

}

sig Credential {}

sig Password {}

sig Payment\_Info {}

sig Location {

latitude: Int,

longitude: Int

} {

latitude >= -90,

latitude <= 90,

longitude >= -180,

longitude <= 180

}

sig Reservation {

user: User,

car: Car,

start\_area: Safe\_Area,

start\_time: Time,

expired: Bool

}

sig Time {

year: Int,

day: Int,

hours: Int,

minutes: Int

} {

year >= current\_year,

year <= 2030, // For some reasons we limit the maximum time interval between the reservation timestamp and its start\_time.

day >= 0,

day < 365,

hours >= 0,

hours < 24,

minutes >= 0,

minutes < 60

}

sig Ride {

reservation: Reservation,

passengers: Int, // User is included in passengers.

pickup\_time: Time,

release\_time: lone Time,

release\_battery\_level: lone Percentage,

release\_area: lone Safe\_Area

} {

passengers >= 1,

passengers <= 4 // We assume that all cars are “small” and equally capient.

}

// General area (e.g. area whose center is the user location, used to find the closest cars).

sig Area {

center: Location,

radius: Int

} {

radius > 0

}

// Area where I can park the car.

sig Safe\_Area extends Area {}

// Area where I can park the car and charge it.

sig Special\_Safe\_Area extends Safe\_Area {}

fact noUsersWithSameCredentials {

all u1, u2: User | (u1 != u2) => u1.credential != u2.credential

}

fact noCarsWithSameIds {

all c1, c2: Car | (c1 != c2) => c1.id != c2.id

}

/\*

TODO:

FACTS:

- A user can have only one reservation each hour.

- A user can be on a ride at a time.

- Company.cars = Car.

- Company.safe\_areas = Safe\_Area.

DON’T WANT TO THINK NOW:

- car.available => the car has a reservation for at most one hour ago or there aren’t any rides whose release\_time is not set yet.

- car.location is in a Safe\_Area in which the last ride with such car finished.

- user.location is near car.location and there’s a reservation with them => car.unlocked = true.

- user.location is not near car.location and there’s a reservation with them and car.unlocked == true => car.unlocked = false. // We want to lock the car if the user goes away because he reminds that he left the gas open!

- car.engine\_on => ride.charge ++.

- after an hour from the reservation.start\_time there are no rides with such reservation => reservation.expired == true and reservation.charge += fee.

- ride.pickup\_time > ride.reservation.start\_time.

- ride.release\_time is empty or ride.release\_time[0] > ride.pickup\_time.

\*/

## Alloy result

## World generated

# Future development

There are a lot of possible improvements in the system to be:

* Accident management

# Used tools

# 

# Hours of work

# 

# Changelog

# Class notes on this project

## Goals

* Should we specify the rewards for virtuous behaviours? General goal: encourage the user to behave well. Rewards and penalties can be seen as subgoals. Therefore it’s NOT necessary to write all rewards and penalties.

## General

* Operators that charge the cars, move the cars (only for point ‘e’).
* Credit card payments done with an external service (that’s a boundary of our system). External services are always system boundaries and external actors in the use cases.
* We must manage all cases that are not in the domain properties/text assumptions.
* We should decide between managing accidents or not (I hope not).
* The payment is always successful => Domain property.