Politecnico di Milano

Software Engineering 2

Requirement Analysis and Specification Document (RASD)

Author:
Claudia Baz
(10916443)

Author:
Pablo Llorente
(10904511)

Author:
David Quintero
(10832704)

Lecturer:
Prof. Matteo Camilli

October 21, 2024

POLITECNICO
MILANO 1863

Contents

1 Introduction

1.1 Purpose

In the last years, Governments' efforts in achieving their emission goals have resulted in the promotion of some incentive schemes to increase electric vehicle (EVs) sales, as they are a crucial means to meet environmental goals. The significant reduction in the prices of battery cells and the 5G rollouts for charging management services are also driving the growth of this market.

However, some factors, such as the lack of infrastructure (charging stations), the still high battery costs, and the multiple vendors with still no standardized charging and payment systems, affect the owners of EVs. As a result, the charging process tends to be time-consuming, forcing drivers to drive around the city searching for a suitable and available socket to charge their vehicles, interfering with their daily schedule.

Therefore, building a system that brings the user closer to the charging stations will improve enormously the user experience. The new system, e-Mall, is composed of two subsystems, e-Mobility Service Provider (eMSP), and Charge Point Management Syst(CPMS). The first faces the users (clients), and the latter faces the charging point operators (vendors), enabling the charging service market. The e-Mall system will allow users to book a charging service, monitor the battery of their vehicles, receive proactive suggestions about where and when to charge, and will ensure the energy supply chain, all from remote.

This document will further expand on the goals and requirements put on the system to be with the purpose of guiding the development.

1.1.1 Goals

G1	The user should have enough information about him/her battery/vehicle status, station locations, socket availability and their prices to make an informed decision about where and when to charge his/her vehicle
G2	The user should be able to book/unbook a socket in a specified charging station and time frame
G3	The user receives personalized suggestions to charge the vehicle based on its battery status, his/her calendar/navigation activities and charging stations information
G4	User should be able to pay to unlock the socket booked to start the charging service and notify him/her when the charging service is finishing and finished

Table 1: Goals

1.2 Scope

The domain of the e-Mall system encompasses all the agents directly involved in the supply and consumption of energy for EVs. The system is composed of two sub-systems, the eMSP and the CPMS:

- The eMSP has interfaces with the User (vehicle owner), the Charge Point Management System (CPMS), the third-party Payment system, the Vehicle itself, and the Calendars of the user. This sub-system allows the user to manage bookings of charging services, monitor the battery status of his/her vehicle, and receive suggestions about the charging schedules to follow based on the user's habits. Thus, making the eMSP a smart system.
- On the other hand, the CPMS has interfaces with the Charging Point Operator (CPO) which we treat as another human agent, the Distribution System Operator (DSO) or energy vendors, the charging stations infrastructure (CSs infra), and the eMSP. Among their functions are to provide automatically the information needed by the eMSP for managing the lifecycle of bookings, ensure the energy supply chain and, enable the charging service execution. Besides that, the CPMS let the CPO to take manually some decisions such as setting the price of charging services, deciding the publishing and creation of special offers, and acquiring energy from the DSOs.

Furthermore, the e-Mall system interacts with the following actors:

- CS Infrastructure: The CS Infrastructure is defined by the set of sockets and batteries in the same location that belongs to a unique CPO. They are managed by the CPMS.
- Human agents
  - User: Schedule bookings.
  - CPO: As it is the operator he decides all about the management of CSs, price of charging services, special offers, and the acquisition of energy.
- Third-party integrations
  - DSO: Energy providers. They sell the energy to the CPOs through the CPMS and provide the estimation of the energy price up to 7 days in advance.
  - Payment system: System used to pay for the charging service.
  - Calendar: App where the user schedules his/her events.
  - Vehicle

Multiplicities and relationships between actors and subsystems

- Each User will have one instance of an eMSP that can pair, if existent, his/her multiple vehicles and calendars.
- There are multiple CPOs and each of them can have only one technological platform CPMS to centrally manage its multiple CSs.
- Each CPMS can interact with multiple eMSPs when broadcasting special offers and on the opposite, each eMSP can interact with multiple CPMSs when searching for sockets available.

1.2.1 World Phenomena

Identifier	Description
WP1	Price of Energy is very fluctuating nowadays (war, geo-politics, conflicts of interest, international situation, source e.g. DSO distance)
WP2	CPO obtain/acquire the Energy from the DSO distribution network or from their batteries (generation/distribution and storing)
WP3	CPO determines the location of the charging stations. It may imply the scarcity of charging stations in some regions/geographies
WP4	Charging station breakdown (e.g. socket or charging cable is broken)
WP5	The user drives to the charging station
WP6	The user charges the vehicle with the socket at the station
WP7	The user updates her/his calendar and uses its navigation system
WP8	CPO interacts with multiple DSOs

Table 2: World Phenomena

1.2.2 Machine Phenomena

Identifier	Description
MP1	The system calculates the shortest path to the selected station
MP2	The system makes queries to an internal database
MP3	The system calculates charging suggestions from user calendar and navigation system activities
MP4	Charging point booking
MP5	The eMPS interacts with multiple CPMSs, each one owned by a different CPO

Table 3: Machine Phenomena

1.2.3 Shared Phenomena

Identifier	Description	Agent
WP1	Control of the CPOs physical Infraesctructure by their associated CPMS.	M
WP2	Payment transaction	M
WP3	The CPMs sends the signal to the CPOs to unlock the skocket after a eMSP registers the payment	M
WP4	User decides the booking detail such as time frame and socket type (charging speed)	W
WP5	User registers or does login	W
WP6	The eMPS notifies the end of the charging	M
WP7	The eMPS shows the charging stations by its location and availability	M
WP8	Vehicle send its battery status information to the system	W
WP9	The system consumes an external API to get information	M
WP10	CPMs calculates the cost of the charging service	W
WP11	CPMs calculates time to fill the full vehicle charge	W
WP12	The eMSP-CPMS system suggests special offers/discounts from the CPOs	M

Table 4: Shared Phenomena

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definitions

Name	Description
Charging Station	Local where vehicles park to charge their batteries. They collect a group of socket types.
Socket	Physical connection/interface where their vehicles connect to charge their battery.
Socket Type	Speed of charge, there are only 3 types: slow/fast/rapid. This is the only difference between sockets.

Table 5: Definitions

1.3.2 Acronyms

Acronym	Name	Description
DSO	Distribution System Operators	Energy vendors
CPMS	Charge Point Management System	CPO management systems. Control their power supply and list their data to external services
CPO	Charging Point Operators	Company or human operator that provides the charging service and is able to acquire energy from DSOs.
eMSP	e-Mobility Service Provider	User interface that provides the interaction with the CPOs through their CPMS
CS	Charging Station	All the infrastructure that is located in the facility where the User charges his/her Vehicle, such as screens to interact with the client, batteries to store energy, and sockets.

Table 6: Acronyms

1.3.3 Abbreviations

Abbreviation	Description
RASD	Requirements Analysis and Specification Document
WP	World Phenomena
SP	Shared Phenomena
MP	Machine Phenomena
GX	Goal number X
DX	Domain assumption number X
RX	Requirement number X
UC.X	Use case number X

Table 7: Abbreviations

1.4 Revision History

RASD 1st version - December 2022

1.5 Reference Documents

References

[1] *Alloy Offical Page*. `https://alloytools.org/`. [Online; accessed 19-December-2022].

[2] *AlloyTools v5.1.0.* `https://github.com/AlloyTools/org.alloytools.alloy/releases/tag/v5.1.0`. [Online; accessed 19-December-2022].

[3] *Requirement Engineering and Design Project: goal, schedule, and rules.* 2022.

1.6 Overview Document Structure

This document is composed of six sections, detailed below.

First, it is introduced the project with its goals. Then, it is presented the scope to determine the general domain of the project, specifying some phenomena covering the system and the world around the system. This first section finishes with some important information: definitions, abbreviations, acronyms...

Second, it is described the system from different points of view. Firstly, some scenarios where users use the system. Then, class and state UML diagrams. Finally, the assumptions made about the domain to simplify the project.

Third, the requirements of the system are specified, including functional, non-functional requirements and external interfaces. Moreover, it is described the use cases and sequence diagrams. Lastly, some mapping between use cases, requirements and scenarios.

The fourth section contains a formal analysis using Alloy.

Finally, it is presented the effort spent by the project members.

2 Overall Description

2.1 Product perspective

2.1.1 Scenarios

1. **User wants to start using the system**

   The user Flavio just bought a new electric car and needs help with the location, prices and any other information of nearby charge stations. He launches the system and starts to sign up, providing all the information needed and finally getting access to the service.

2. **User charges the car**

   The user Gabriella arrives at a charge station. She has previously booked an appointment so 10 min before the time scheduled, while driving, enters the system, logs in and pays to unlock the socket. At the moment he arrives, waits until the time comes, connects the socket to the car and confirms in the service that the car is actually charging. Once she confirms it, checks the time estimation for charging and decides to get a coffee in a nearby bar. She forgets about the time, but, once the charge is full, a notification informs her, pays the coffee, returns to the car, draws the socket and leaves the charge station.

3. **User wants to book an appointment**

   The user Matteo is planning a long trip through the country. So, he needs to charge the car the exit day to have 100% battery on the car so he can reach his destination. First, he enters the service, logs in and checks how much battery is left entering "Monitor Battery Status". Then, he adds the percentage of battery he wants, the date and hour and searches the charging stations on his way to exit the city. Finally, he decides one of them due to its location, selects the slow socket, because he will have time enough before the trip and it is cheaper and confirms the information given to finish the booking.

4. **User modify appointment**

   The user Enzo has previously booked an appointment in a charging point, but, that day a few hours earlier, his little daughter got sick, and he needed to bring her to the hospital, while waiting her hour, he realizes that he won't be able to reach the appointment, so enters the service, logs in, enters "My Bookings" and modifies the date for the next day, when we expects to have more time to charge the car.

5. **User is suggested a special offer**

   The User Niccolo is driving his car through the city. Some days ago, he charged his car, so he does not need to charge it now, because it has 70% of battery. However, he gets informed by the system of a special offer at one charge station near him. He checks the offer, realizes he could save a lot of money if he charges now, instead of a few days later. Selects the indicated charge station and the desired type of socket and confirms the booking.

6. **User is suggested to charge due to schedule**

   The user Valentina is planning a road trip through Italy's north of the country and adds all her planned cities to visit on her personal calendar. Once done, the system suggests a new charging schedule, including charge stations and type of socket aligning with the events inserted by the user in her calendar. At this moment, Valentina reviews all the possible bookings, changes one type of socket due to economical reasons and confirms all bookings.

7. **User is suggested to charge due to low battery**

   The user Leonardo is driving his car exiting the city, outside it there is not any charging station. He did not check his car's battery and when it falls to the 10% the system notifies the user. Then the user, once he knows he needs to charge the car. Checks nearby stations, selects the desired type of socket and confirms the booking.

8. **User pairs the calendar and the vehicle**

   The user Andrea just got registered in the system and continues with the next step. He enters the "Profile" section and pairs his calendar, logging in the service. Then, he enters on "Profile" again and starts with the process of pairing his vehicle, using a 4 digit code that the vehicle shows in the control screen. After pairing both things, Andrea now can plan a trip on its calendar to get suggested a charging schedule or check the vehicle status at any moment.

2.1.2 Class diagram

We use the class diagram to represent the domain model of our system. Here, we show the different classes that encompass sub-systems and actors, their attributes, operations, and the relationships between them.

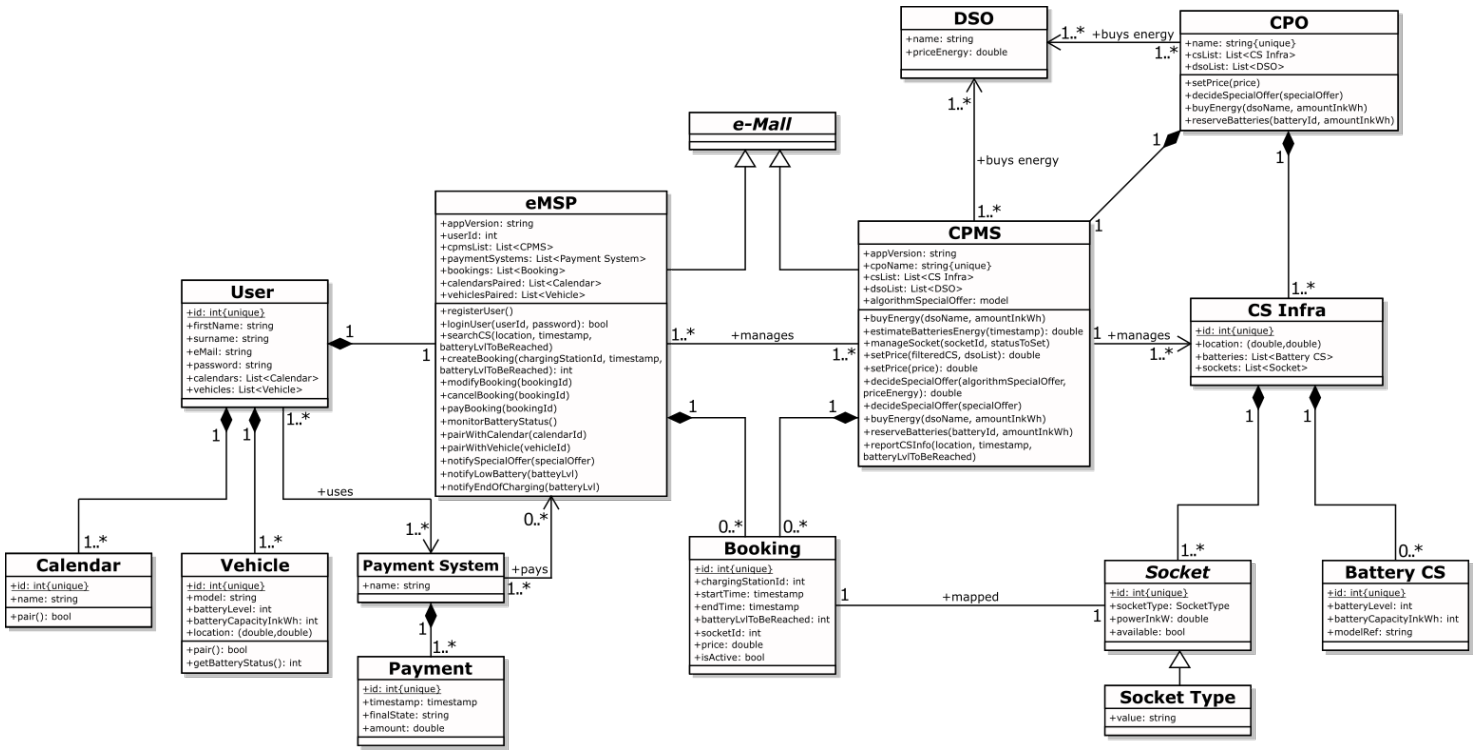

Figure 1: Class diagram

2.1.3 Statechart diagrams

Statechart diagrams are a tool to model the dynamic nature of a system. They describe its state-dependent behavior. In our case, as the system is composed of

two major sub-systems, namely eMSP and CPMS, we decided to show how the system evolves from global state to global state and then zoom in to each global state, depicting the state of both sub-systems and the possible interactions between them.

In this case, it is presented two main state diagrams, with the two main functions of the system: booking and, payment and charging. Moreover, there is a state diagram "Collect charging stations information" that belongs to Booking as all the states that the CPMS go across to show the charging station map to the user.

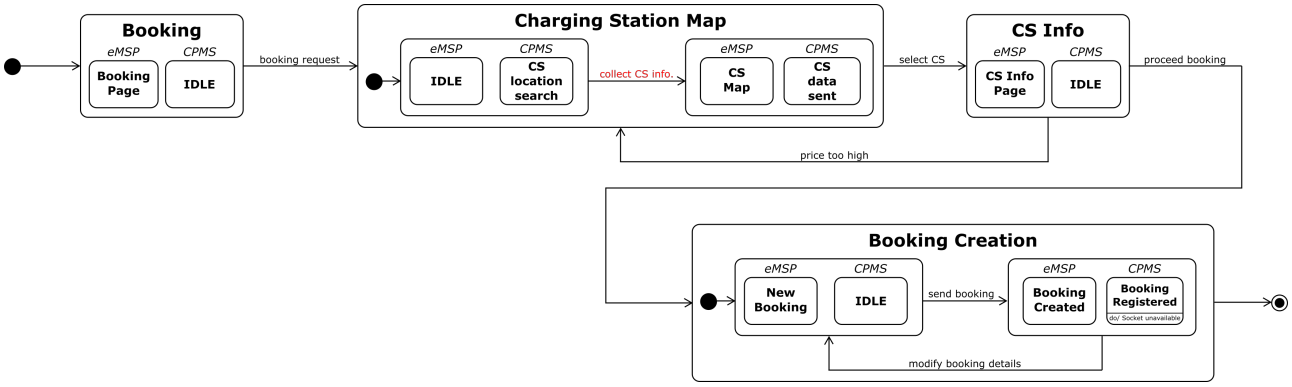

Figure 2: Booking state diagram

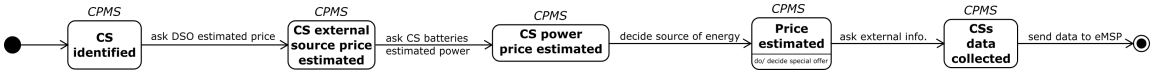

Figure 3: Collect charging station information state diagram

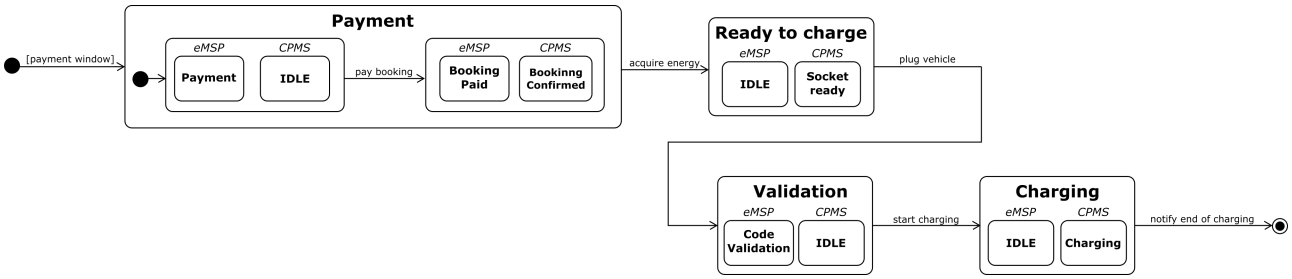

Figure 4: Payment and charging state diagram

2.2 Product functions

The main functions of the e-Mall system are:

2.2.1 Management of a booking for a charging service

One of the most crucial functions of the e-Mall system is the capacity to handle the different aspects associated with the management of a booking. The User can manage all matters related to his/her booking through the eMSP, such as the creation, modification, and cancellation of a booking for a charging service.

Once the User is registered and logged in to the eMSP, he/she can search for a charging station according to certain criteria such as location, booking time, and percentage to be reached. The eMSP sends the booking request to all CPMSs located near the area specified in the search. After that, every CPMS retrieves all the information regarding the external status of its different charging stations (CSs) and assigns the price for the charging service. Namely, the external status of a CS (number of charging sockets available, their type such as slow/fast/rapid, and, if all sockets of a certain type are occupied, the estimated amount of time until the first socket of that type is freed). Next, the User decides the CS, socket type, and time that best suits to him/her, and the CPMS accepts it, thus confirming the reservation.

Once a booking has been created, a User can modify it by deferring it up to 7 seven days after the current day he/she enters this option or canceling it.

2.2.2 Enabling energy supply chain

Another important function of the e-Mall system is to ensure the availability of energy at the charging station at the moment the charging process starts.

There are two possibilities to ensure the energy supply either automatically by the CPMS or manually by the CPO:

- **CPMS:** Once the CPMS received a booking request, it estimates the energy in the batteries at the time of the booking start and, in the same way, asks to the DSOs for the estimation of the energy price for that day. Once it has this data it decides from which source to obtain the energy and associates the amount of energy to retrieve from each source for that booking. The CPMS makes it available by reserving the energy from its batteries and/or acquiring energy from the DSO selected only after the User has paid the booking.
- **CPO:** The CPO is free to decide when and how much energy to buy from the DSOs. To that end, he can make use of the CPMS or contact the DSO by any other means.

2.2.3 Charging process execution and monitoring

The e-Mall system makes possible the actual charging process by enabling the use of the socket only to the user that booked it during a time slot.

Once the User has paid the booking and has plugged the Vehicle into the assigned socket, he/she asks the eMSP to generate the 4-digit code needed to unlock the CS to make use of the socket and enters it into the CS Infra. Then, the CPMS sends the code to the eMSP of the User and this validates if the codes match. After that, the charging process starts and the User can monitor the battery status at any moment through the eMSP. Apart from that, the eMSP is constantly monitoring the battery level of the Vehicle and sends a notification to the User when the battery has reached the 90% or 100% of the desired battery to be reached. For example, if the User asks to charge it till 80% then he/she will receive two notifications about the end of charging, one at 72% and another at 80%.

2.2.4 Smart system - Proactive suggestions

The e-Mall system has some features that make it a smart system such as recommendations based on patterns identified in the daily routine of the User or some specific events.

The eMSP sub-system is able to suggest some recommendations to the User thanks to the possibility of pairing with the calendars and the vehicles of the User. Thus, the eMSP has access to the GPS and battery information of the Vehicle, the location of the device where the app is running, and information about the events the User plans to attend that have been added to the Calendar. With all these data, the eMSP can detect some patterns in the daily routine of the User such as the most visited places, events he/she usually attends, preferred charging schedule, and CSs, among others. In that way, the eMSP can suggest booking schedules to the User.

2.3 User characteristics

The following actors are considered in the e-Mall system:

- **Unregistered User**
  Owner of a vehicle that needs to register to the eMSP platform before being able to use any of its functionalities. He/she wants to avoid wandering around the city searching for an available socket, eliminating in that way the time devoted to that task.

- **Registered User**
  Owner of a vehicle that has the same needs as the Unregistered User, but in contrast, he/she is registered to the eMSP platform and thus is able to book a charging service, modify and cancel it. Apart from that, the User can monitor the battery status of his/her vehicle and receive some suggestions from the system. He/she updates is calendar constantly.

- **CPO**
  Representative person of the CPO that uses the CPMS sub-system to set

the price of the charging service, publish special offers and acquire energy from the DSOs. These actions can be handled either manually by a CPO agent or automatically by CPMSs.

Registered User and CPO are both registered to different platforms, their corresponding sub-system, but both interact with the e-Mall system as a whole.

2.4 Assumptions, dependencies, and constraints

2.4.1 Domain assumptions

Identifier	Description
D1	There is always 100% of compatibility between the vehicle hardware and the device where the eMSP is running
D2	The charging stations physical sockets have an universal adaptor for all kinds of vehicles. Meaning, the only differences between types of sockets is the charging speed and all cars can physically connect to all sockets.
D3	When a socket is not being used it is consider always available assuming it wont have any physical defect or missfunction
D4	All users leave the station after the battery charge has reach the expected percentage
D3	When a socket is not being used it is consider always available assuming it wont have any physical defect or missfuntion.
D5	No user disconnects the vehicle from the socket before the expected time
D6	All users arrive an pay on time their booking
D7	The interaction between the various providers (eMSPs, CPOs, and DSOs) occurs through uniform APIs *
D8	All the eMSPs detect automatically the CPMSs available and when one disconnects
D9	The energy acquired from the DSO is available immediately at the charging station
D10	The estimations of energy and price of charging are very exact, they don't fluctuate with time so the user is not charged twice
D11	The DSO can provide an estimation of the energy price up to 7 days in advance
D12	The CS batteries have infinite capacity, they can buy as much energy as they can afford, with no restriction in storage capacity.
D13	All users configure their app properly, pairing at least one calendar and one vehicle

Table 8: Domain Assumptions

*Even though D7 this might be consider as an implementation detail, the project document given [3] proposes it as as assumption. Therefore, all the interactions and interfaces between different components are based on it.

Goal	Domain assumptions
G1	D1, D4, D7, D8, D10, D11, D13
G2	D3, D4, D6, D7, D8, D11, D13
G3	D7, D8, D10, D13
G4	D1, D2, D3, D5, D6, D7, D8, D9, D10, D11

Table 9: Domain assumptions x Goals Mapping

3 Specific Requirements

3.1 Internal Interface Requirements

The CPMS-eMSP system is divided in two different subsystems. The eMSP, that is the entry point for the user input and the CPMS, that must act as a middle man between the eMSP, the CPOs, their CSs and the DSOs.

3.1.1 Integrator

Each CMPSs provides a personalize service for its CPOs and CSs. Therefore, even though they provide uniform functionalities, each one represents a different entry point with individual attributes. In that manner, there must be an integrator to connect the eMSP system with the diverse CPMSs. Its aim must be to provide bidirectional services between the eMSP and the CPMSs.

3.2 External Interface Requirements

The eMSP and the CPMS systems can not analysed individually. They depend on shared external services to consume and retrieve data. To fit all their useCases, they must be able to interact with them through interfaces.

3.2.1 eMSP external interfaces

User Interface The eMSP is the final client interface. Its aim is to serve a graphical interface where the user can interact with the charging stations and his/her vehicle data. It must be able to define the correct inputs to the user to consume and retrieve data to the CPMSs. Besides that, it should be available in mobile devices and in the car system (screen where the user interacts).

Hardware Interface The eMSP must be able to interact with the vehicle battery sensor to monitor its status. In that manner, should be an universal interface between the application and the car.

3.2.2 CPMS external interfaces

The CPMS system act as a server-side applications, meaning, they should not have direct contact with the user. Due to the D7 in table 2.4.1, the next

interfaces are assume to be between softwares:

Communication Interface - CS The CPMSs must be able to interact with the CS hardware to monitor the charging process, unlock/lock them by eMSP requests and manage the CS energy supply. In that manner, the CPMS don't interact directly with the client vehicle.

Communication Interface - CPO The CPMS must be able to let the CPO modify or set manually the prices and the special offers calculated automatically. Besides that, the CPO must be able to acquire energy through the CPMS to manage the CS energy supply. In that manner, it is an interaction between software and operator.

Communication Interface - DSO While the DSOs supply energy to the CPOs, the CPMS act as an intermediary between them both. It makes decisions about which DSO will supply the energy or if there is no need for that. In that manner, it is an interaction between the CPMS and the DSO software.

3.3 Requirements

3.3.1 Functional Requirements

Identifier	Description
R1	The eMSP interface shall allow the user to select a charge station from a list based on its location
R2	The eMSP interface shall allow the user to select an available socket from a charge station list with its socket types, availability and prices
R3	The CPMS must be able to identify and list the location of a charging station, the number of its available and unavailable sockets, their type (charge speed), their cost, and, the estimated amount of time until the first socket of a type is freed
R4	The eMSP shall allow the user to book freely only available sockets based on a determined timeframe and a desired percentage of battery to be reached,
R5	The CPMS must change the socket status to unavailable the timeframe it has been booked
R6	The eMSP must notify the CPMS to start charging the vehicle only if the user booking starts and has paid for energy to be consumed
R7	The eMSP shall allow the user to cancel a book
R8	The eMSP must provide a graphical interface where the user can consume information about the battery status of his/her vehicle
R9	The eMSP must not let a user to book two sockets with overlapping times
R10	The eMSP must collect user charging habits such as the mean charging time of a book, frequented charging stations, and type of sockets used.
R11	The CPMS must be able to list the DSOs' energy prices
R12	The CPMS must be able to set the charging price, create special offers and acquire energy automatically
R13	The CPMS must be able to let the CPO set the charging price, create special offers and acquire energy manually
R14	The CPMS must be able to know the amount of energy available in the charging stations batteries and decide the source of energy: either from a DSO, the stations internal batteries or a mix of both
R15	The eMSP shall filter the offers listed by CPMS relevant to the user (is consider relevant, if its booking price is lower than the usual, considering the type of socket the user tends to select and the frequented charged stations)
R16	The eMSP shall notify the user when the vehicle battery has only 10 and 5 percent of charge
R17	The CPMS must monitor the amount of power supplied by the socket when a vehicle is charging to infer when the battery has reach the expected percentage and the time left to end of the charge
R18	The eMSP must interact with the CPMS charging monitoring process to notify the user when the charging service has been 90% and 100% completed
R19	The eMSP must monitor the vehicle when its driving, turn on and off to keep the user informed about its charge
R20	The eMSP shall allow the user to pay before the charging service
R21	The eMSP shall allow the users to do login and register

Table 10: Functional Requirements

3.3.2 Non Functional Requirements

Identifier	Description
R22	The eMSP must have an interface to interact with the vehicle (hardware or software) to compile information about its battery status
R23	The eMSP must provide an authentication to identify their users securely and to save their credentials in the device: automatic login
R24	The eMSP must have a mobile push notification system to send information about the battery status of the vehicle and charging stations with offers and lower prices
R25	The eMSP must interact synchronously with the CPMSs for the booking feature
R26	The eMSP must consume the CPMSs charging station information every time the user opens its interface to show updated information
R27	The eMSP must be able to interact with a third party payment system
R28	The CPMS must be able to interact with the physical hardware of the CSs infrastructure to unlock the sockets
R29	The eMSP must be able to consume data from the calendar/navigation system of the user
R30	The eMSP must have an interface integrator to interact with multiple CPMSs from different CPOs

Table 11: Non Functional Requirements

Goal	Functional Requirements
G1	R1, R3, R4, R5, R8, R11, R12, R13, R14, R17, R19, R21
G2	R1, R2, R3, R4, R5, R6, R7, R9, R17, R19, R21
G3	R1, R3, R10, R11, R12, R13, R14, R15, R16, R17, R19, R21
G4	R1, R5, R6, R17, R18, R20, R21

Table 12: Functional Requirements x Goals Mapping

Goal	Non Functional Requirements
G1	R22, R23, R26, R30
G2	R23, R25, R26, R30
G3	R22, R23, R24, R29, R30
G4	R23, R24, R27, R28, R30

Table 13: Non Functional Requirements x Goals Mapping

3.4 Use cases

1. **Register User**

Subsystem	eMSP
Actor	Non-Registered User
Entry conditions	The User does not have an account and is on the eMSP homepage.
Event flow	1. The User presses "Register" button. 2. The User enters name, email address and password. 3. The User clicks on "Create account" button. 4. The eMSP sends confirmation by e-mail of the successful registration.
Exit condition	Account was created successfully
Exceptions	The User does not specify all the data needed for creating the account. The eMSP warns the User about the missing data and stays on the same screen till the User submits the data successfully.

Table 14: UC.1 - Register User

2. **Login User**

Subsystem	eMSP
Actor	Registered User
Entry conditions	User is registered, not logged in and on the eMSP homepage.
Event flow	1. User presses login button. 2. User enters email address and password. 3. User clicks on "Login" button. 4. eMSP authenticates the User and displays a success message.
Exit condition	User is logged in into the eMSP app.
Exceptions	1. User do not enter password or email before submitting. 2. User enters invalid email/password combination In all cases eMSP will notify the User.

Table 15: UC.2 - Login User

3. **Pair eMSP with Vehicle**

Subsystem	eMSP
Actor	Registered User, Vehicle
Entry conditions	The User is already registered, logged in, and is on the homepage of the eMSP.
Event flow	1. The User presses the "Profile" button. 2. The User presses the "Pairing" button and then the "Pairing with Vehicle" option. 3. The eMSP shows a pop-up requesting the User to connect the Vehicle to the same network as the one it is connected. 4. The User does that and presses "Continue". 5. The eMSP generates a 4 digit code and sends it to the vehicle. 6. The Vehicle shows a 4 digit code in its control screen. 7. The eMSP asks the User for that code. 8. The User inputs the code and presses "Pair" button. 9. The eMSP validates if the code matches. 10. The eMSP shows a "Pairing successful" message.
Exit condition	The eMSP is paired with the Vehicle.
Exceptions	The User enters a wrong code, the eMSP shows a "Wrong code" message and requests the User to start the pairing procedure again from scratch.

Table 16: UC.3 - Pair eMSP with Vehicle

4. **Pair with Calendar**

Subsystem	eMSP
Actor	Registered User, Calendar App
Entry conditions	The User is already registered, logged in, and is on the homepage of the eMSP.
Event flow	1. The User presses the "Profile"" button. 2. The User presses the "Pairing"" button and then the "Pairing with calendar"" option. 3. The eMSP shows Google Calendar, Microsoft Outlook Calendar, and Apple Calendar as options for calendars. 4. The User selects one calendar service. 5. The eMSP invokes the API of the corresponding service. 6. The User logs in to the calendar service. 7. The User authorizes the eMSP to manage the calendar.
Exit condition	The eMSP can manage the User's calendar, i.e. see and create events
Exceptions	The User could not log in to the calendar service, the eMSP notifies this to the User and invite him/her to try again.

Table 17: UC.4 - Pair eMSP with Calendar

5. **Book a timeslot**

Subsystem	eMSP
Actor	Registered User, Calendar App
Entry conditions	The User is already registered, logged in, and is on the homepage.
Event flow	1. The User presses "Book a charging station"" button entering in the filter selection view. 2. The User inputs the dates, desired battery's charge percentage to be reached, tentative time, and place where he/she wants the charge to take place. The eMPS can only show information for a maximum of 7 days from that day. If the place is not given, it is used the same as the current position given by the GPS of the device. 3. The eMSP asks all nearby CPMSs for the information of each charging station (socket). The information is: location, type of sockets available, estimated cost and time of charging per socket based on the desired battery's charge percentage to be reached, etc. 4. **UC.7 - Report charging stations information.** 5. The User checks different options of charging stations and selects the socket type from his/her preferred one. 6. The User confirms the booking hours or selects a different one and clicks on "Confirm booking"" button. 7. The eMSP verifies that there is no overlap with other active bookings. 8. The eMSP sends the booking details to the CPMS 9. The eMSP adds this event to the User's calendar. 10. **UC.6 - Manage socket status**
Exit condition	The booking is added to the User's calendar. The socket is provisionally set to busy during the booked timeframe. The User has to pay the booking the same day it is effective.
Exceptions	1. User inserts a percentage lower than the current battery level. The eMSP warns the User about this, so he/she can either confirm or correct the inserted data. 2. The User does not specify a place and the GPS is turned off. The eMSP warns about this and asks the User to either specify the place or to turn on the GPS. 3. The User tries to select a date more than 7 days onwards, the eMSP warns the User about the impossibility of making a booking for those days and invites him/her to modify the date."

Table 18: UC.5 - Book a timeslot

6. **Manage socket status**

Subsystem	CPMS
Actor	CS Infrastructure
Entry conditions	The User has booked a socket from the charging station during a timeslot or has canceled the booking. The CPMS receives the booking details such as socket type, location, and timeslot.
Event flow	1. If booking request: The CPMS sets a socket to unavailable during the timeframe specified in the booking. 2. If cancel booking request: The CPMS sets a socket to available during the timeframe specified in the booking.
Exit condition	The socket is provisionally set to busy during the booked timeframe.

Table 19: UC.6 - Manage socket status

7. **Report charging stations information**

Subsystem	CPMS
Actor	No direct interaction with actors, only through other use cases.
Entry conditions	The eMSP has asked all nearby CPMSs for the information of each charging station (socket), i.e. external- and internal CS info, and the price of the booking.
Event flow	1. **UC.8 - Check the external status of charging stations.** 2. **UC.9 - Set price of charging.** 3. The CPMS collects all the information.
Exit condition	Info about all the nearby stations to the position specified is assembled.
Exceptions	If there is no socket available, skip the set price of the charging step.

Table 20: UC.7 - Report charging stations information

8. **Check the external status of charging stations**

Subsystem	CPMS
Actor	Charging Station Infrastructure
Entry conditions	The eMSP asked for the information of a charging station.
Event flow	1. The CPMS retrieves from the sockets their location, the external status of the charging station, i.e. number of charging sockets available, their type such as slow/fast/rapid, and, if all sockets of a certain type are occupied, the estimated amount of time until the first socket of that type is freed. 2. The CPMS collects all the information of the charging station. 3. The CPMS sends all the required info to the eMSP."
Exit condition	The CPMS reports the "external" status of a charging station.

Table 21: UC.8 - Check the external status of charging stations

9. **Set price of charging**

Subsystem	CPMS
Actor	CPO, DSO
Entry conditions	The eMSP asked for the information of the charging stations during a booking process or CPO decides to set the price manually.
Event flow	1. The CPMS estimates the amount of energy available in the batteries of its charging station at the moment of the charging process. 2. The CPMS retrieves the estimated price of the energy from the DSOs. 3. The CPMS decides the source of the energy to provide the service: DSO, own batteries, or a mix of both. 4. The CPMS or CPO estimates the cost of the charging service.
Exit condition	The CPMS or CPO decided the source of the energy, the price of the charging service, and the special offer (if possible).

Table 22: UC.9 - Set price of charging

10. **Decide special offer**

Subsystem	CPMS
Actor	CPO
Entry conditions	The CPMS or CPO determined the price for a charging service.
Event flow	1. Based on a certain rule or algorithm, the current energy market state, and after knowing the current price of the energy, the CPMS or CPO determines whether to set a special offer or not. 2. The CPMS broadcasts the offer to all eMSP.
Exit condition	The CPMS or CPO set or not a special offer. In a positive case, it broadcasts the offer to all nearby eMSPs.

Table 23: UC.10 - Decide special offer

11. **Notify special offer available**

Subsystem	eMSP
Actor	Registered User
Entry conditions	The CPMS set a special offer and broadcasted it to all eMSP e.g. 30% discount if the charge exceeds X kWh
Event flow	1. The eMSP receives this special offer and depending on the charging habits of the User, his/her location, and other details decides whether to recommend it or not to him/her. 2. If yes, the eMSP sends a notification to the User about this special offer.
Exit condition	The User receives a notification of a special offer.

Table 24: UC.11 - Notify special offer available

12. **Modify booking**

Subsystem	eMSP
Actor	Registered User, Calendar App
Entry conditions	The User is already registered, logged in, and is on the homepage. The User wants to modify the booking but keep the same charging station.
Event flow	1. The User presses the "My Bookings" button entering the menu with all the future bookings. 2. The User presses the booking to be modified. 3. The User clicks on the "Modify" button and enters the booking modification view. 4. The User modifies the date and starting time of the booking, the battery's percentage to be reached, and the socket type. The date of the new reservation cannot be scheduled for more than 7 days from the current day on which the modification is being made. If the User wants a different charging station, he/she should cancel the booking and create a new one. 5. The eMSP asks the specific CPMS of the booking for the information on the charging station. 6. **UC.7 - Report charging stations information.** 7. The User selects the new starting time and clicks on the "Confirm" button. 8. The eMSP verifies that there is no overlap with other active bookings. 9. The eMSP modifies the User's calendar according to the updated booking details. 10. The eMSP sends the updated booking details to the CPMS. 11. **UC. 6 - Manage socket status.**
Exit condition	The modified booking is added to the User's calendar. The socket is provisionally set to busy/unavailable during the booked timeframe. The User has to pay for the booking the same day it is effective.
Exceptions	1. The User clicks on the charging station ID trying to change it. The eMSP warns the User about the impossibility to do so on this screen, suggesting canceling the booking and creating a new one.

Table 25: UC.12 - Modify booking

13. **Cancel booking**

Subsystem	eMSP
Actor	Registered User, Calendar App
Entry conditions	The User is already registered, logged in, and is on the homepage. A booking is existent.
Event flow	1. The User presses “My Bookings” button entering the menu with all the future bookings. 2. The User presses the booking to be canceled. 3. The User clicks on the “Cancel” button. 4. The eMSP prompts a window to confirm the cancellation. 5. The User clicks on “Confirm” button. 6. The eMSP sends the updated booking details to the CPMS and deletes the event from the User’s Calendar. 7. The CPMS set the socket as available during the just deleted booking timeframe.
Exit condition	The charging station is now set to available for future bookings in the same timeframe

Table 26: UC.13 - Cancel booking

14. **Pay booking**

Subsystem	eMSP
Actor	Registered User, Payment System
Entry conditions	The User is already registered, logged in, and is on the homepage of the eMSP. He/she has already booked a charge using the eMSP app, it is within the payment time window, and he/she proceeds to pay for the booking.
Event flow	1. The User presses "My Bookings" button entering the menu with all the future bookings. 2. The User presses the booking to be paid. 3. The User clicks on the "Pay" button 4. The eMSP checks if there are 20 minutes or less left for the start of the reservation and redirects the User to the platform of the External Payment System. 5. The User inserts his/her online financial credentials and gets authenticated. 6. The User proceeds with the payment. 7. The eMSP sends the payment signal to the CPMS and enables the "Start charging" button on the app.
Exit condition	The User is allowed to use the Charging Station Infrastructure during the booked timeframe.
Exceptions	1. The reservation period expired during the payment attempt. The eMSP warns the User about this and invites him/her to book again but with the appropriate adjustment in the starting time. 2. The User enters the wrong credential. The external payment system manages it and invites the User to try again. 3. The User attempts to pay outside the payment time window. The eMSP prompts a screen saying "Try again when there are less than 20 minutes left for the start of the reservation".

Table 27: UC.14 - Pay booking

15. **Acquire energy**

Subsystem	CPMS
Actor	CPO, DSO, Charging Station Infrastructure
Entry conditions	The User has paid for the charging service or CPO decides to acquire energy manually.
Event flow	1. CPMS or CPO reserves the energy to be used from the batteries. 2. CPMS or CPO acquires the energy to be used from the DSO.
Exit condition	Enough energy is available to start the charging process.

Table 28: UC.15 - Acquire energy

16. **Start charging process**

Subsystem	eMSP, CPMS
Actor	Registered User, Vehicle, Charging Stations Infrastructure
Entry conditions	The User has already done the booking and paid for it so the "Start charging" button is enabled. He/she is in the charging station, in front of the booked socket and already plugged the Vehicle into it. He/she is on the homepage.
Event flow	1. The User presses the "My Bookings" button entering the menu with all the future bookings. 2. The User presses the booking to be started. 3. The User clicks on the "Start charging" button. 4. The eMSP prompts a 4-digit code to be entered into the charging station. 5. The User enters the code and presses "Start" in the charging station Infra. 6. The Charging Station Infrastructure sends the User's input to the CPMS. 7. The CPMS receives the most recently generated code from the eMSP. 8. The CPMS validates if the code entered by the User matches with the generated by the eMSP. 9. The charging process starts.
Exit condition	The charging station charges the Vehicle.
Exceptions	If the CPMS doesn't detect the connection to the Vehicle, it asks to the User to do it or do it properly.

Table 29: UC.16 - Start charging process

17. **Monitor battery status**

Subsystem	eMSP
Actor	Registered User, Vehicle
Entry conditions	The User is already registered, logged in, and is on the homepage of the eMSP. The Vehicle is paired with the eMSP.
Event flow	1. The User presses the "Monitor battery status" button. 2. The eMSP asks the Vehicle sensor for the current battery level of the Vehicle. 3. The Vehicle sends the battery data to the eMSP. 4. The eMSP shows the battery level data and: i. If the Vehicle is charging, it shows the estimated time left till the desired battery percentage is reached. ii. Otherwise, it only shows the current battery status.
Exit condition	The User can monitor the current battery status.

Table 30: UC.17 - Monitor battery status

18. **Notify end of charging**

Subsystem	eMSP
Actor	Registered User, Vehicle
Entry conditions	The User has already started the charging process of his/her Vehicle and left the charging station facility.
Event flow	1. The eMSP monitors the charging process by asking about the battery status of the Vehicle. 2. Once the battery has reached the 90% or 100% of the desired percentage level, it sends a signal to the User notifying the soon end or indeed end, respectively, of the charging process.
Exit condition	The User receives two notifications about the end of the charging process.

Table 31: UC.18 - Notify end of charging

19. **Notify battery level is low**

Subsystem	eMSP
Actor	Registered User, Vehicle
Entry conditions	The battery level of the Vehicle is in 10% or 5%. The eMSP is paired with the Vehicle and thus can monitor its battery status
Event flow	1. The eMSP detects that the battery level of the Vehicle has reached 10% or 5%. 2. The eMSP notifies the User of the current battery level and suggests him/her charge the Vehicle promptly.
Exit condition	The User receives notifications about the low battery level of the Vehicle, one when it reaches 10% and the other when it reaches 5%.

Table 32: UC.19 - Notify battery level is low

20. **Suggest charging schedule**

Subsystem	eMSP
Actor	Registered User, Calendar App
Entry conditions	The User has updated his agenda on the Calendar with a road trip that starts in less than 7 days. The User has paired the eMSP with his/her calendar and is logged in.
Event flow	1. The eMSP detects the new event in the Calendar. 2. The eMSP designs a charging schedule i.e. charging stations where to stop, type of socket to use based on the expected time to arrive at the next charging station, the overall duration of the event, the socket availability, and the estimated percentage level of the battery during the trip. The eMSP assumes that the Vehicle is fully charged at the beginning of the trip. 3. **UC.7 - Report charging stations information**, but the CPMS chooses the charging stations. 4. The eMSP sends the suggested charging schedule to the User. 5. The User checks the plan and reviews each possible booking in the suggested charging stations. 6. The User does the corresponding modifications. 7. The User approves the bookings.
Exit condition	The User has booked the needed charging stations in order to meet his/her agenda.
Exceptions	Part of the event takes place 7 days from now. Since the eMSP only suggests booking for a period of 7 days from now, it warns the User about it and recommends scheduling the rest of the bookings individually or splitting the event into smaller events

Table 33: UC.20 - Suggest charging schedule

Mapping Use Cases on Requirements

Use case ID	**Use case name**	**Requirements**
1	Register User	R21
2	Login User	R21,R23
3	Pair eMSP with Vehicle	R22
4	Pair eMSP with Calendar	R29
5	Book a timeslot	R1,R2,R4,R9,R15,R25,R26
6	Manage socket status	R5,R28
7	Report charging stations information	R3,R11,R12,R13,R14,R17
8	Check the external status of charging stations	R3,R16
9	Set price of charging	R3,R11,R12,R13,R14,R15
10	Decide special offer	R11,R12,R13,R14,R15
11	Notify special offer available	R24
12	Modify booking	R1,R2,R4,R9,R15,R25,R26
13	Cancel booking	R7,R25
14	Pay booking	R20,R25,R27
15	Acquire energy	R12,R13,R14
16	Start charging process	R6,R28
17	Monitor battery status	R8,R24
18	Notify end of charging	R18,R19
19	Notify battery level is low	R16,R19
20	Suggest charging schedule	R10,R15,R29

Table 34: Mapping use cases on requirements

Scenarios	**Use cases**
User wants to start using the system	UC.1,UC.2
User charges the car	UC.2,UC.14,UC.16,UC.18
User wants to book an appointment	UC.2,UC.5,UC.6,UC.7,UC.8,UC.9,UC.17
User modify appointment	UC.2,UC.6,UC.7,UC.8,UC.9,UC.12
User is suggested a special offer	UC.2,UC.10,UC.11
User is suggested to charge due to schedule	UC.2,UC.5,UC.6,UC.7,UC.8,UC.9,UC.20
User is suggested to charge due to low battery	UC.2,UC.5,UC.6,UC.7,UC.8,UC.9,UC.19
User pairs the calendar and the vehicle	UC.2,UC.3,UC.4

Table 35: Mapping scenarios on use cases

3.4.1 Use Case Diagram

A use case diagram is a graphical representation of all the possible interactions between a system and its users and helps to depict the scenarios of interactions, goals that the system helps the actors to achieve, and the scope of the system itself.

In the following use case diagram, we consider the User to be registered and logged in for all use cases owned by the eMSP different from "Register User" and "Login User". Moreover, we used a different colour for each external actor to improve the diagram comprehension.

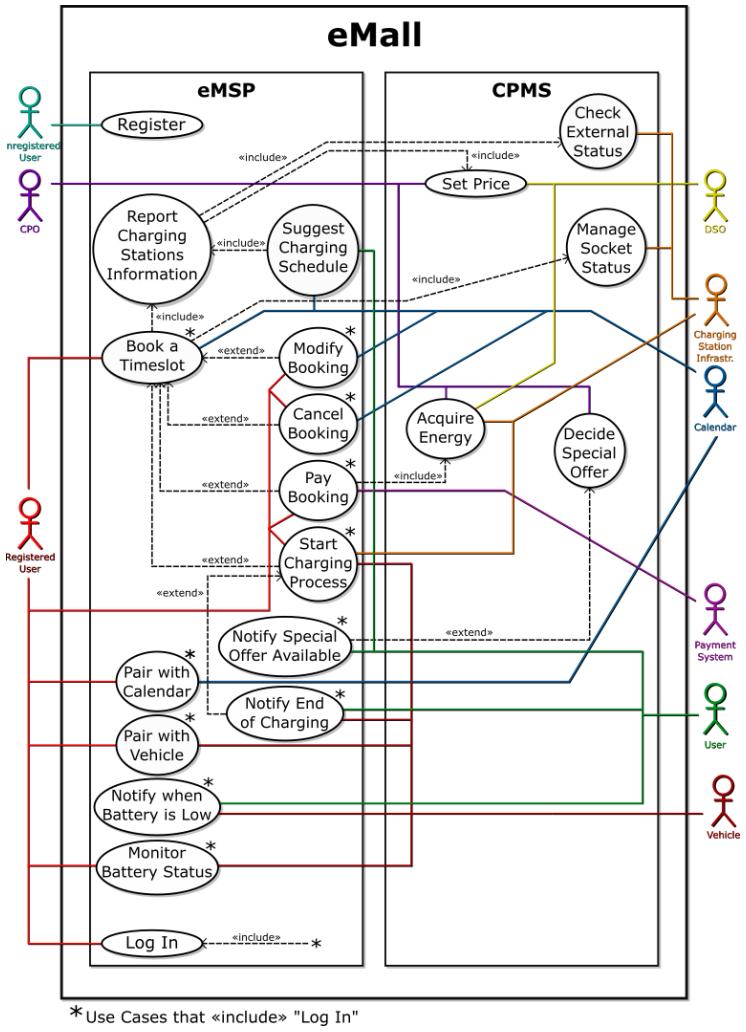

Figure 5: Use Case diagram

3.5 Sequence diagrams

A sequence diagram models the set of events/messages sent between objects of a system and the actors (a.k.a external agents) that interact with it in a single use case.

In this section, the corresponding sequence diagrams for the use cases are

presented. As in the use case diagram, we consider the User to be registered and logged in for all use cases owned by the eMSP different from "Register User" and "Login User". This is in order to simplify the sequence diagrams and make them more understandable. Therefore, some of the use cases are a part of other use cases, those are coloured in red and its initial and final events are not duplicated, they are specified in the general use case or in the specific one, but never in both, so it is possible to follow the flow of events through all this sequence diagrams.

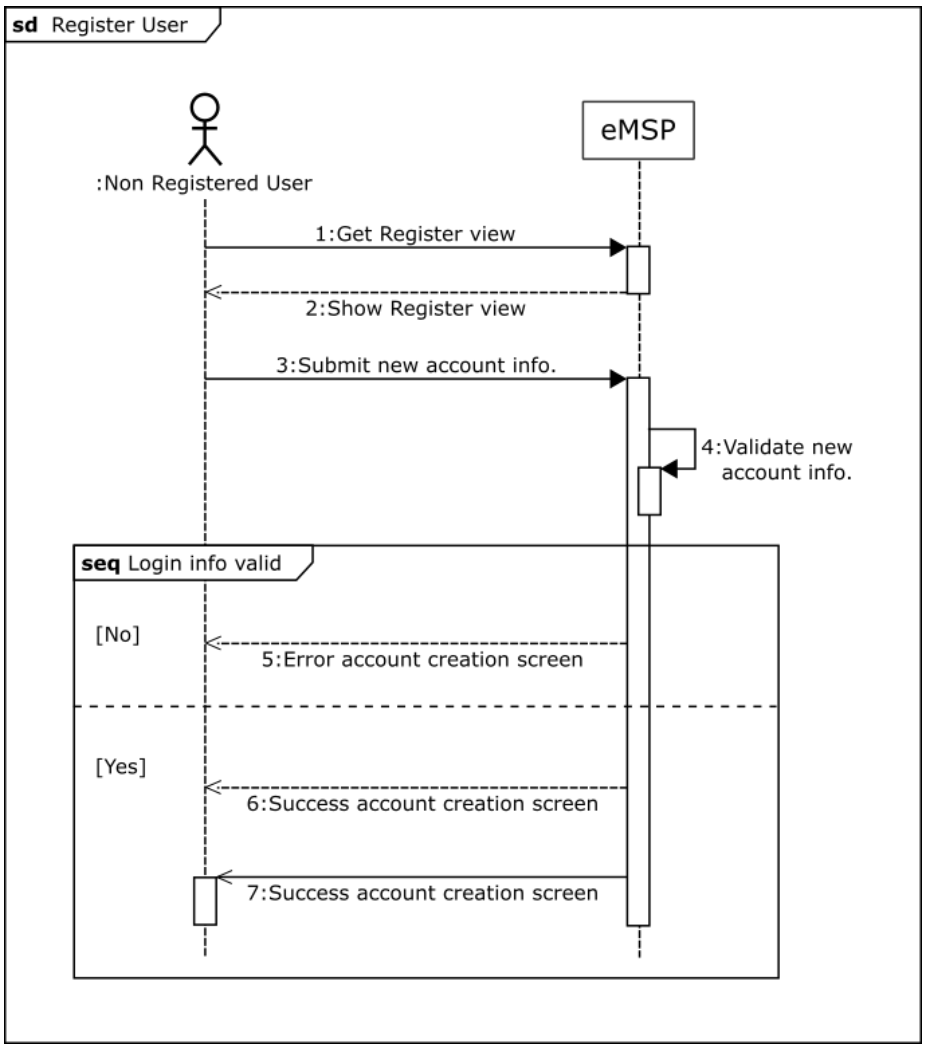

Figure 6: UC.1 - Register User

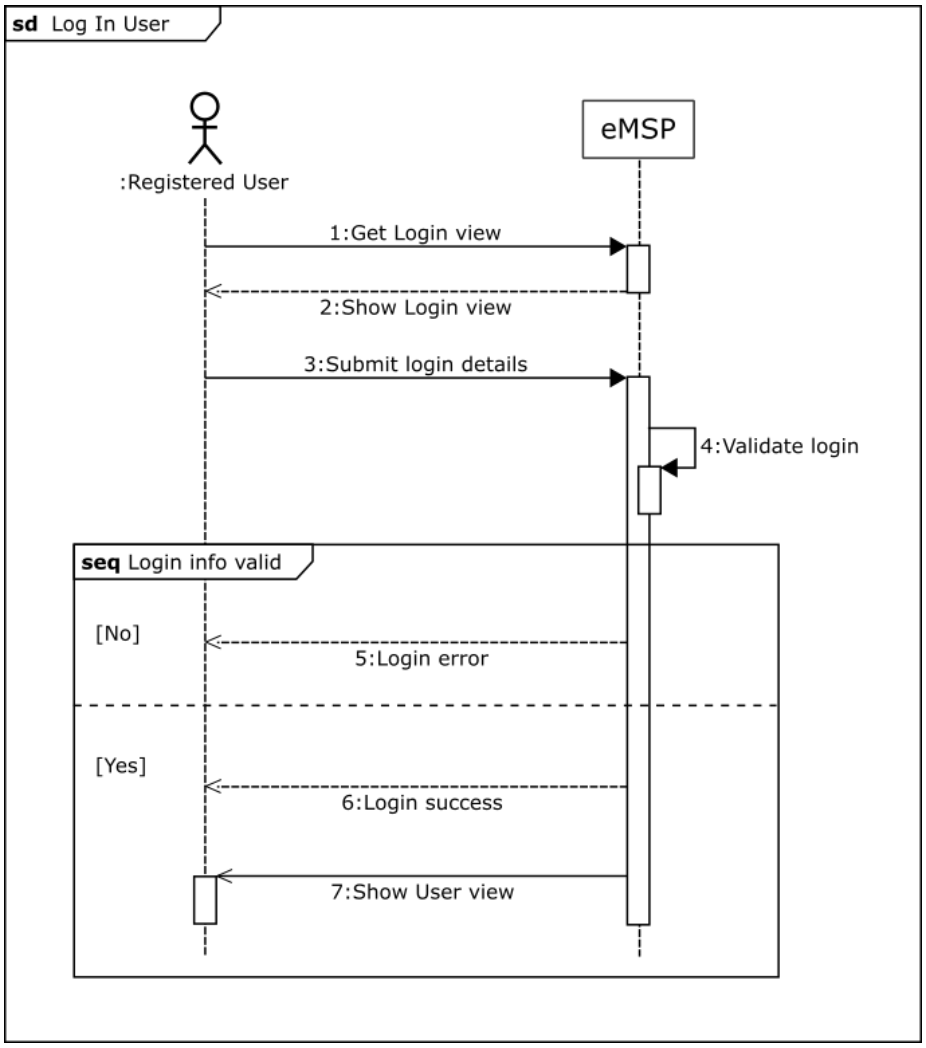

Figure 7: UC.2 - Login User

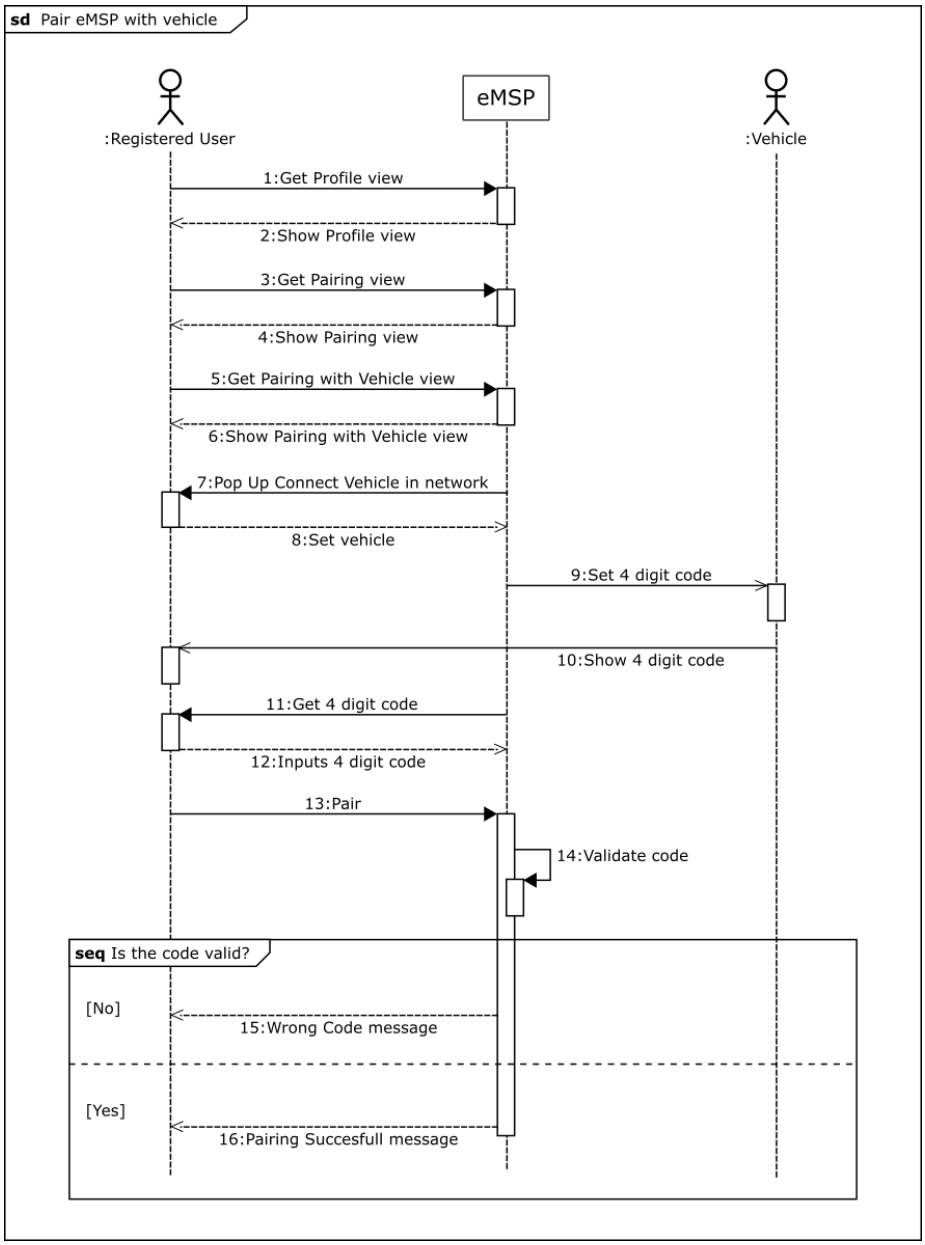

Figure 8: UC.3 - Pair eMSP with Vehicle

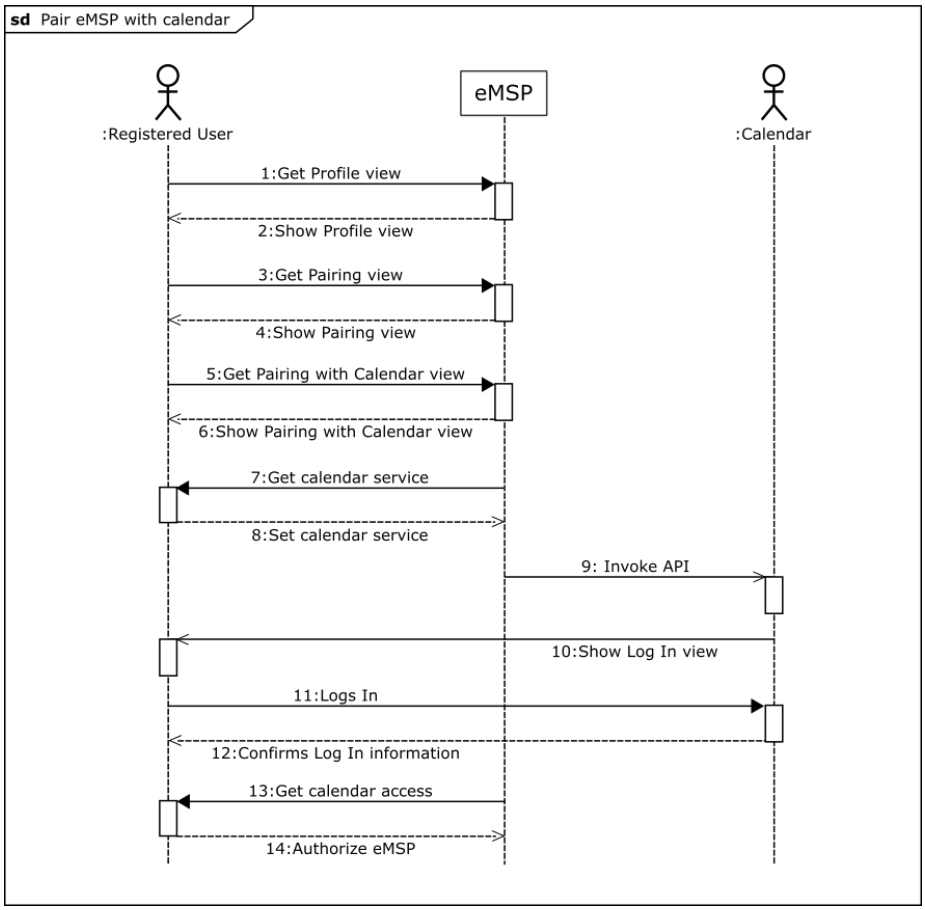

Figure 9: UC.4 - Pair eMSP with Calendar

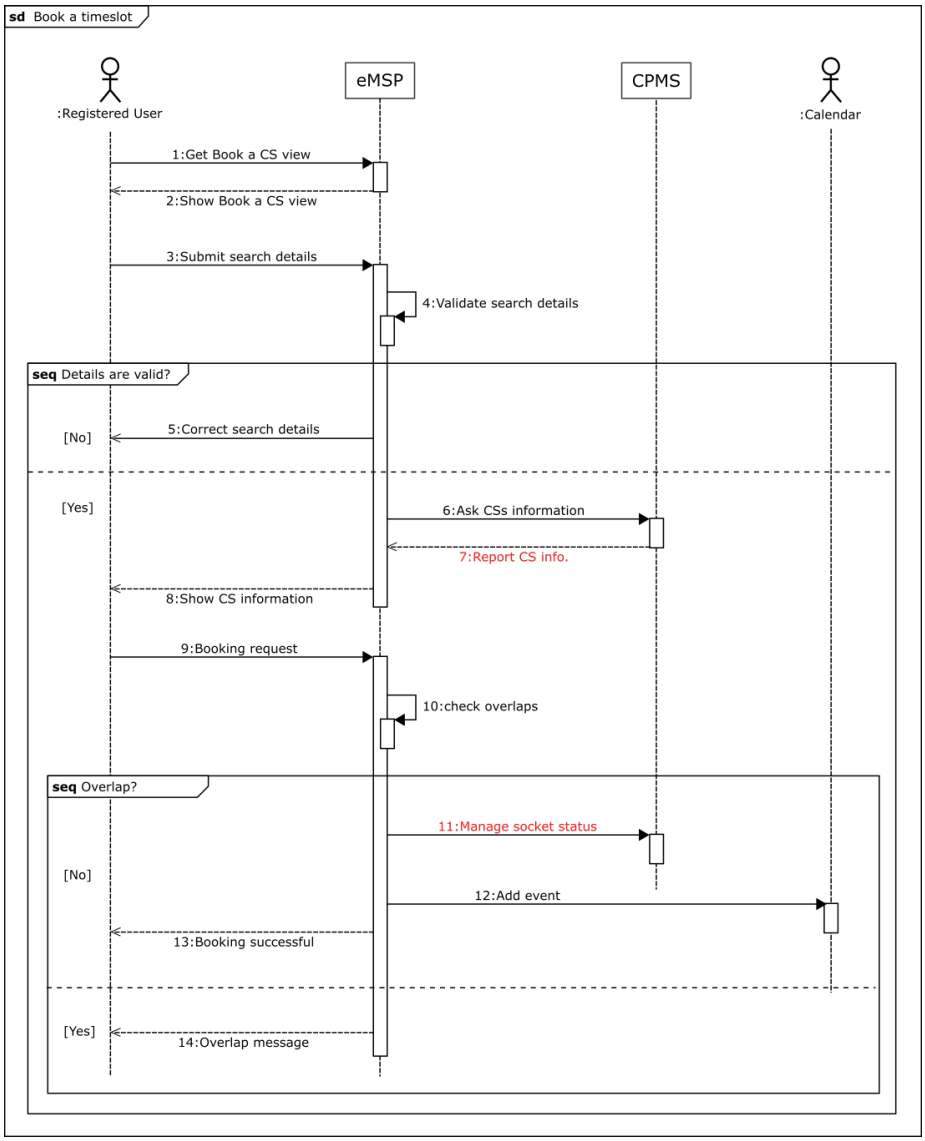

Figure 10: UC.5 - Book a timeslot

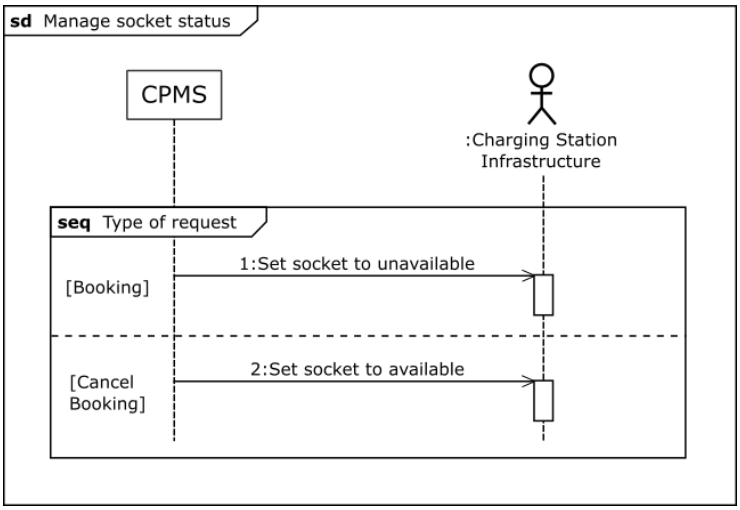

Figure 11: UC.6 - Manage socket status

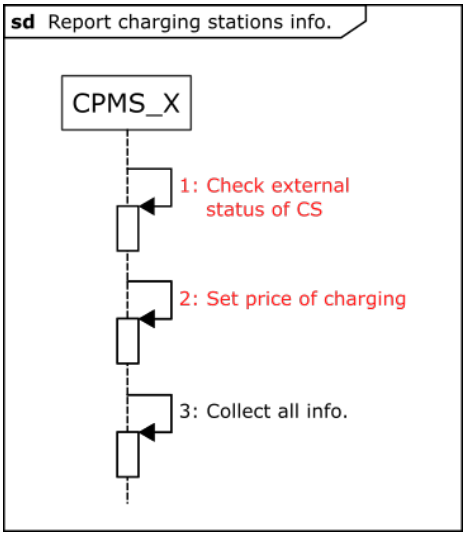

Figure 12: UC.7 - Report charging stations information

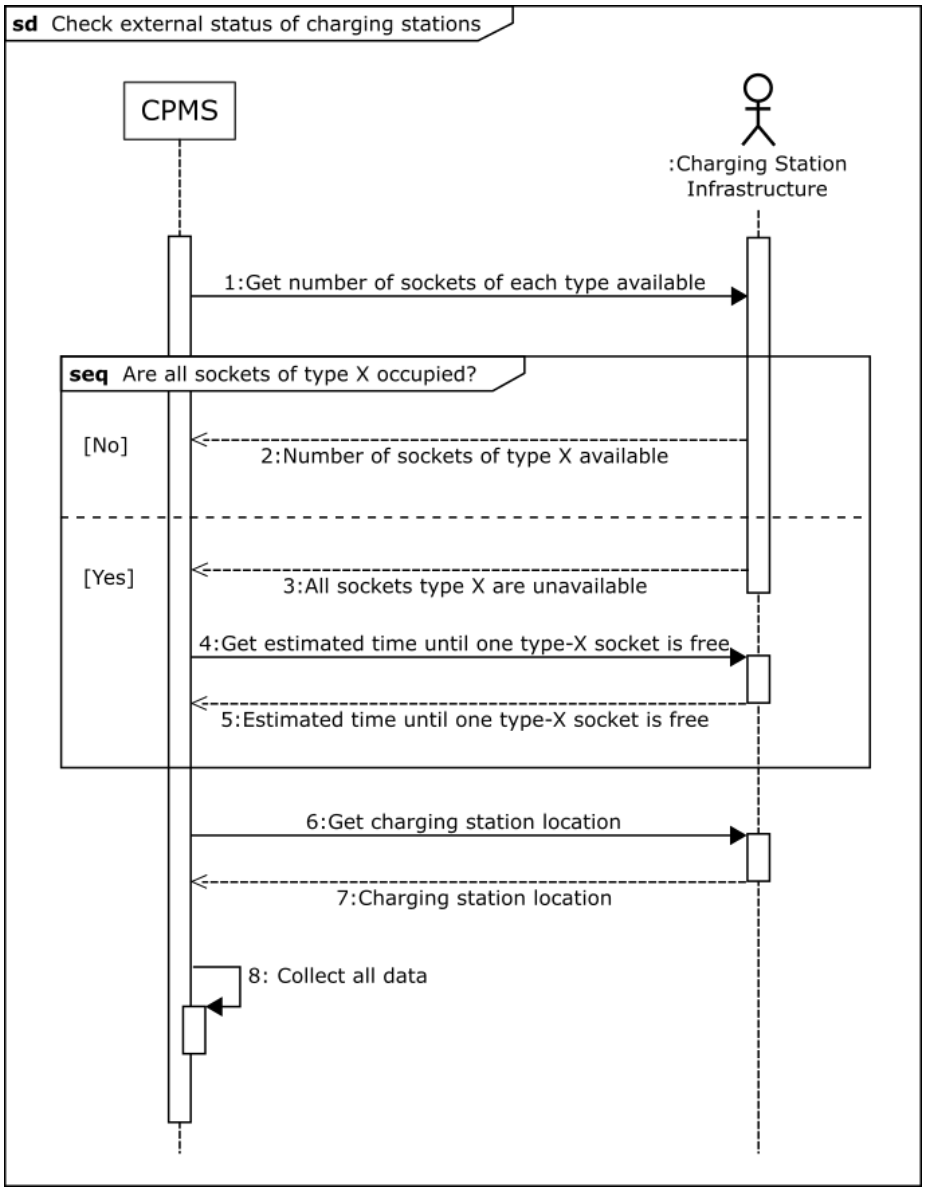

Figure 13: UC.8 - Check external status of charging stations

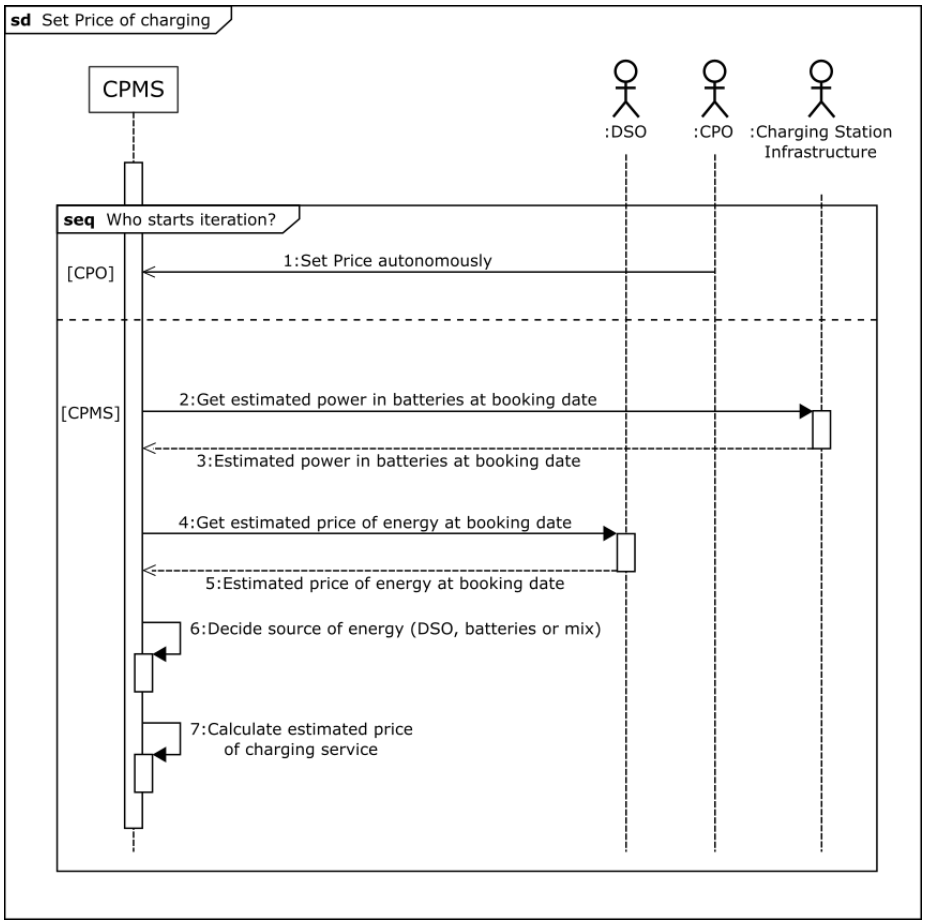

Figure 14: UC.9 - Set price of charging

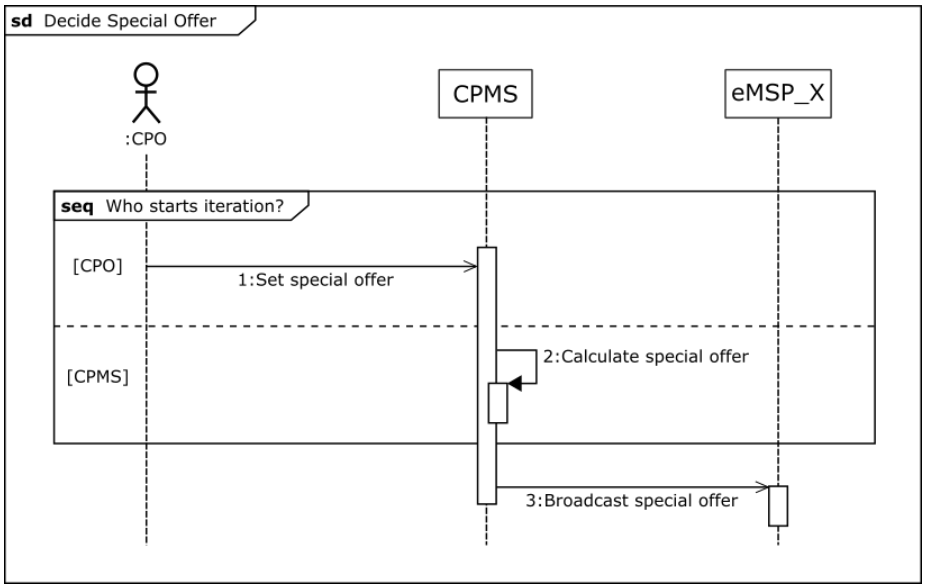

Figure 15: UC.10 - Decide special offer

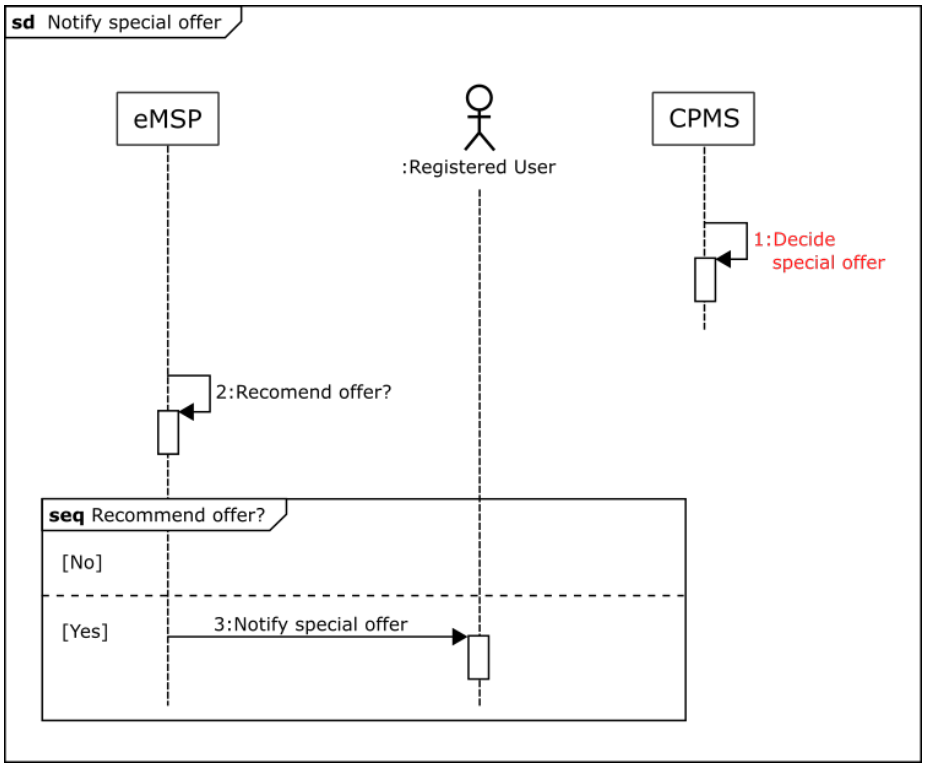

Figure 16: UC.11 - Notify special offer available

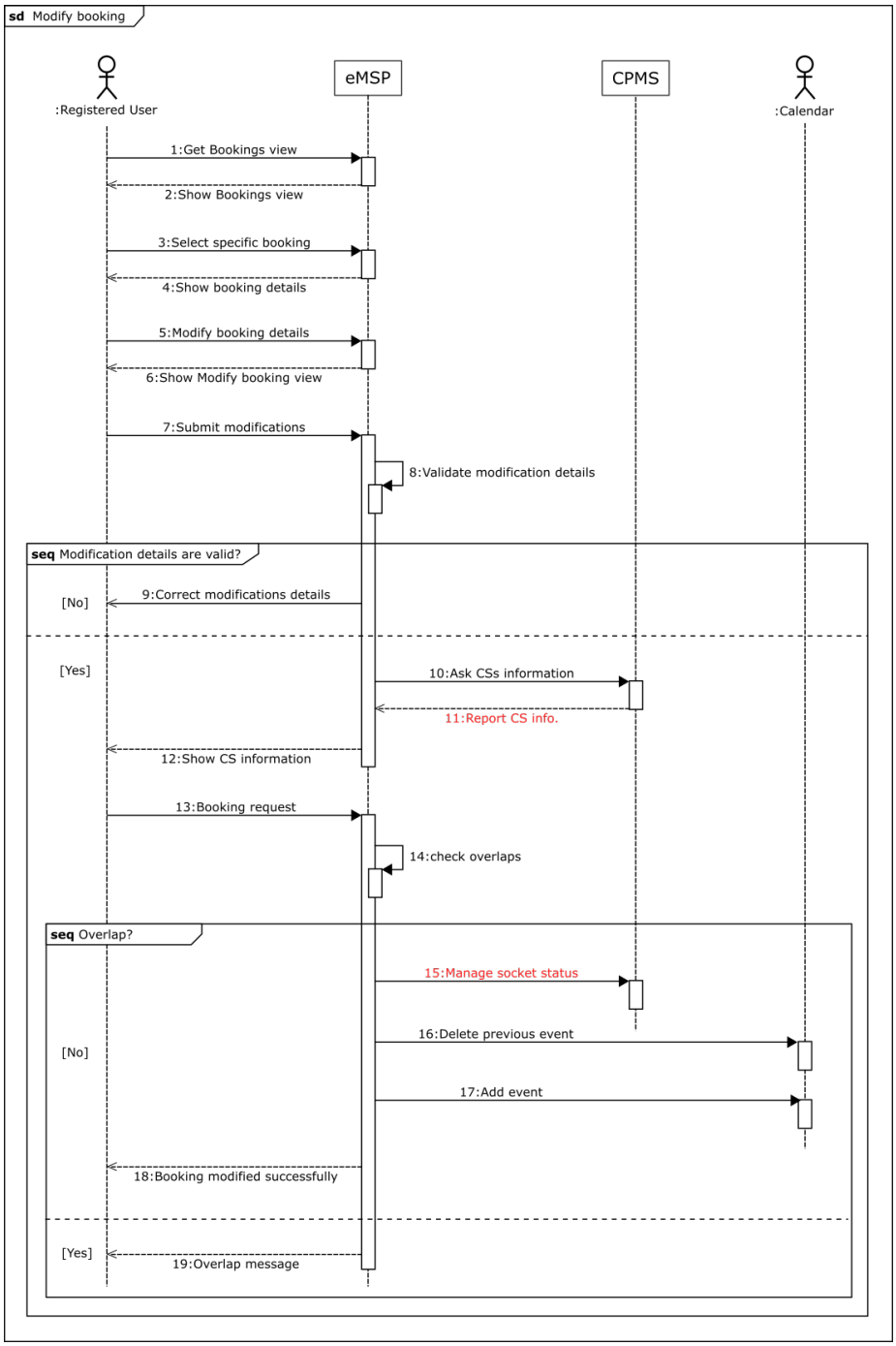

Figure 17: UC.12 - Modify booking

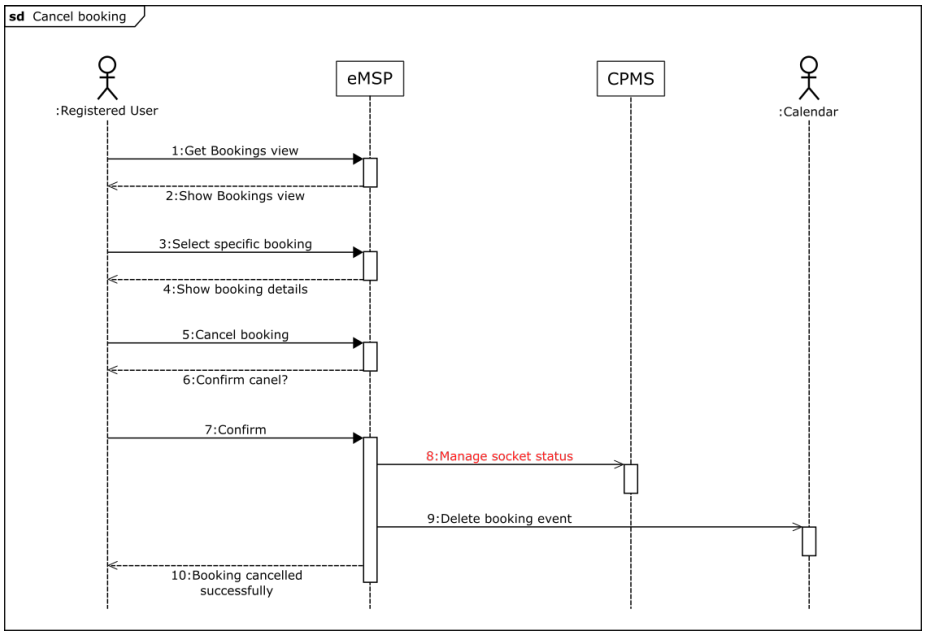

Figure 18: UC.13 - Cancel booking

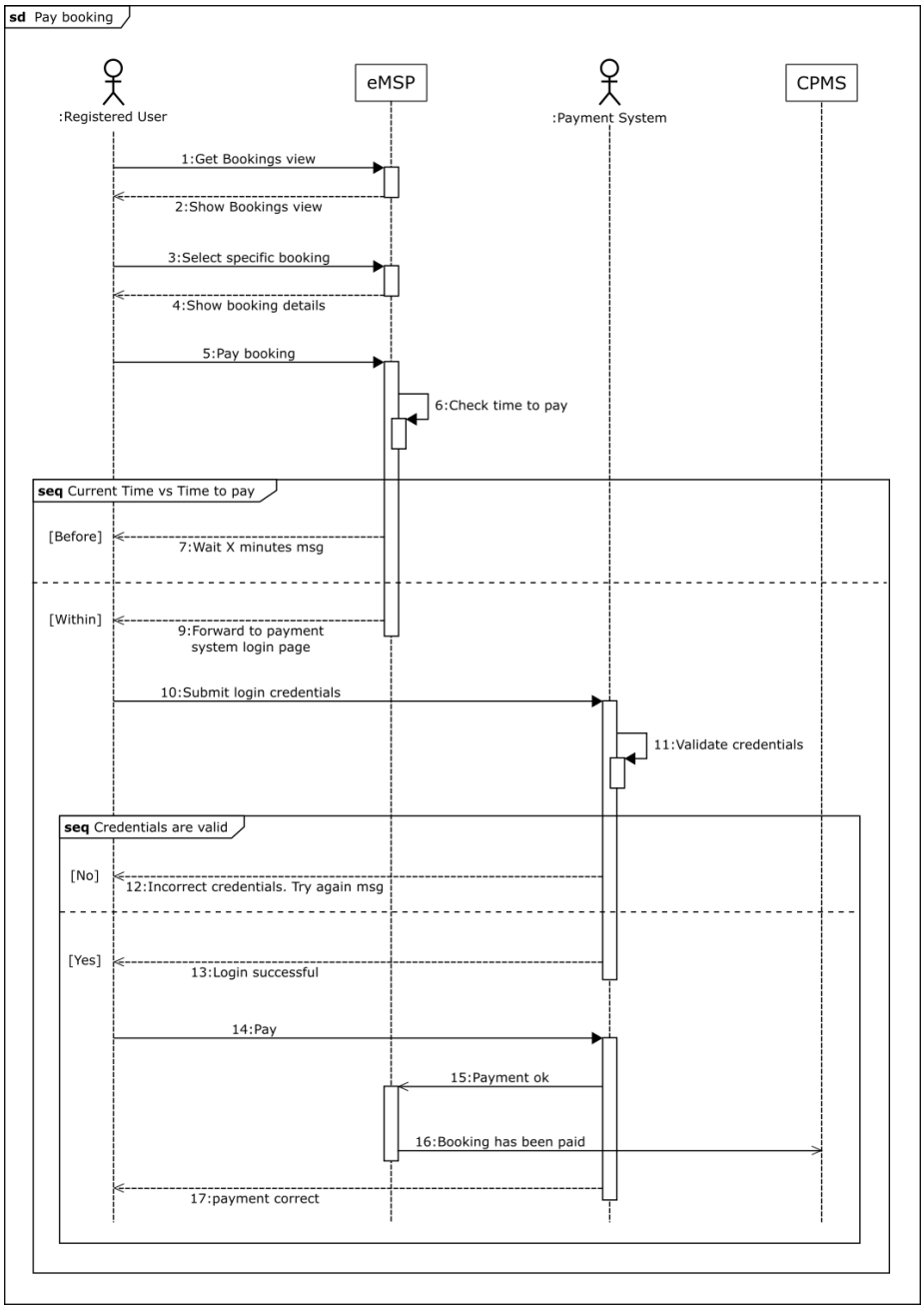

Figure 19: UC.14 - Pay booking

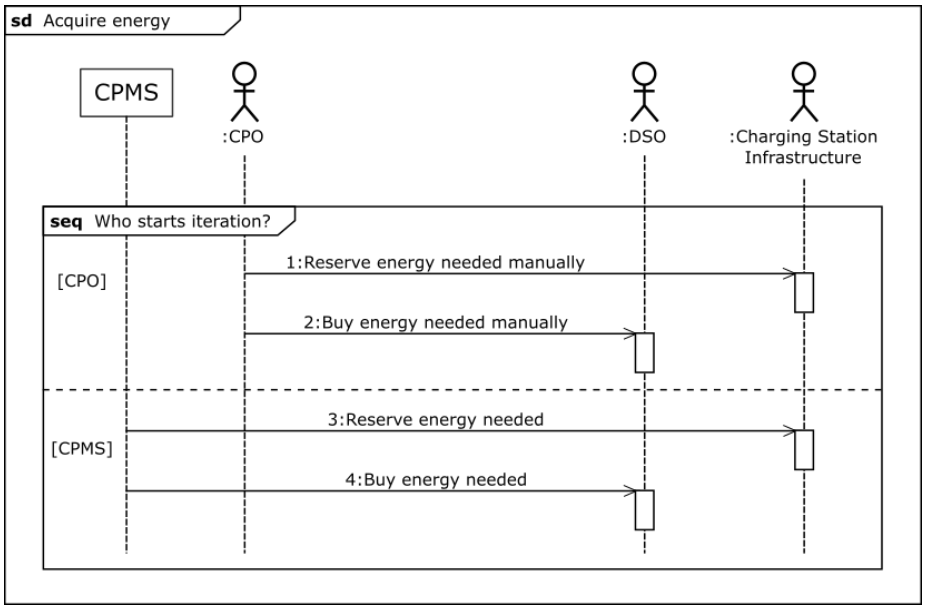

Figure 20: UC.15 - Acquire energy

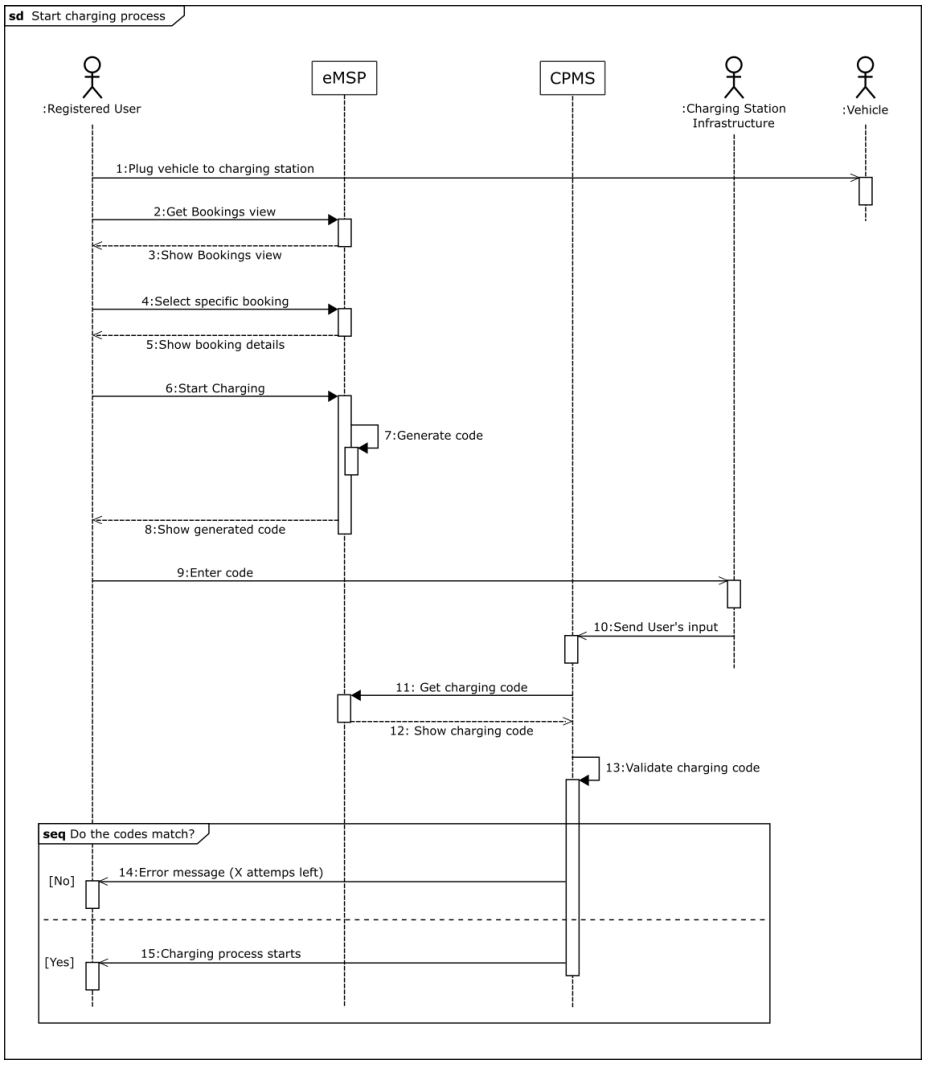

Figure 21: UC.16 - Start charging process

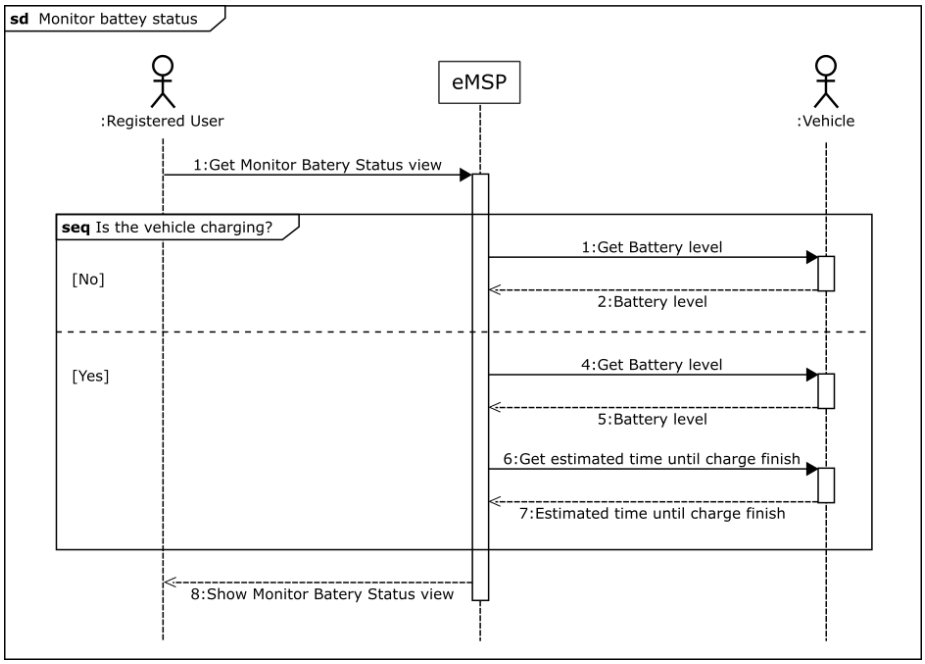

Figure 22: UC.17 - Monitor battery status

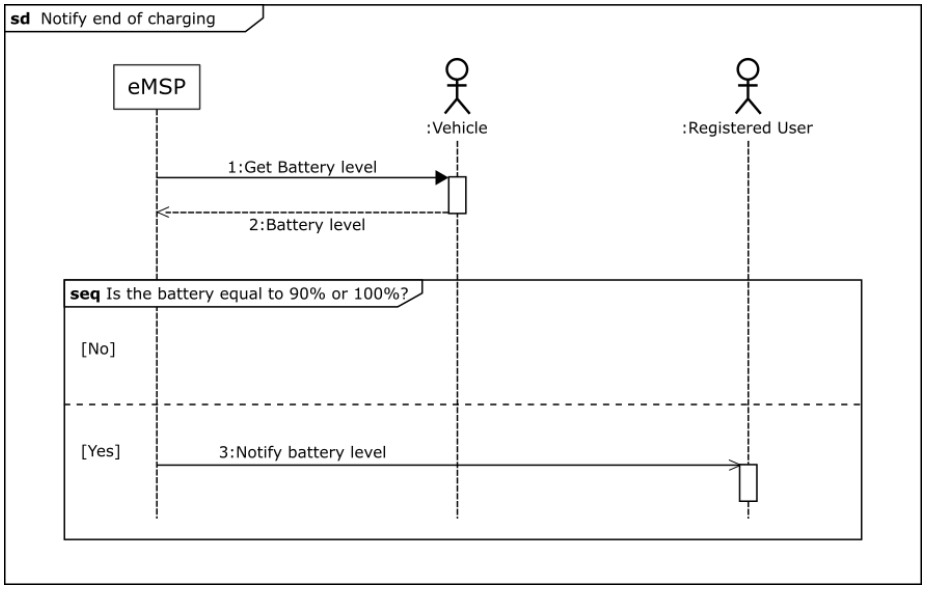

Figure 23: UC.18 - Notify end of charging

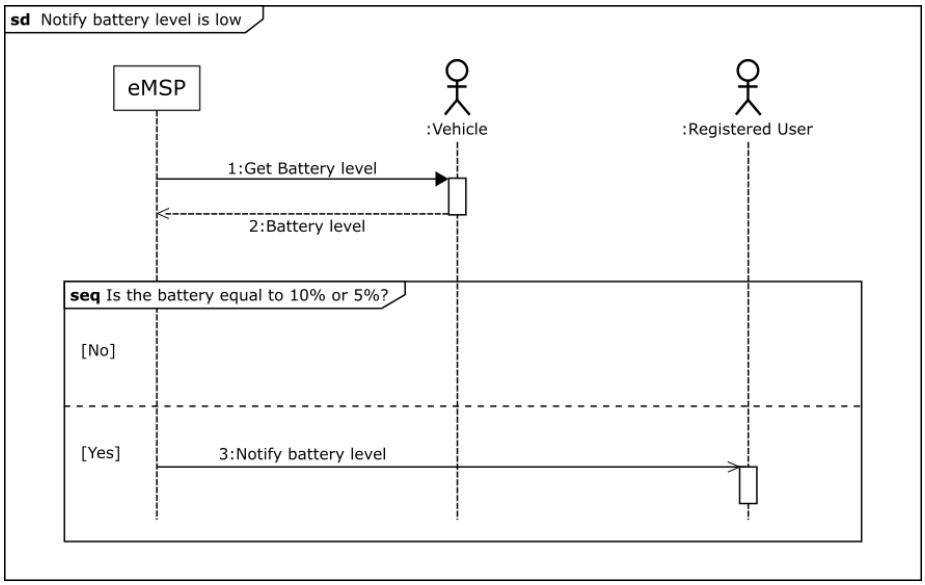

Figure 24: UC.19 - Notify battery level is low

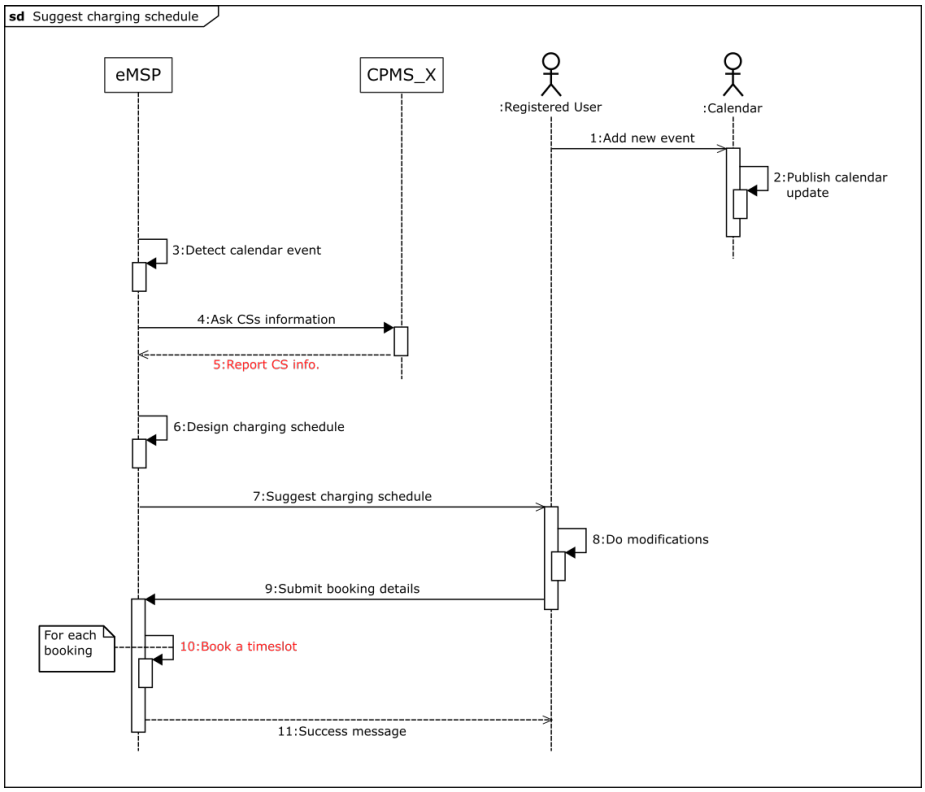

Figure 25: UC.20 - Suggest charging schedule

3.6 Performance Requirements

The eMSP application is meant for user devices, therefore it shouldn't have high processing requirements. Its aim is more focused on consume and sending data. In that manner, it is expected Internet access.

The application can be use in several scenarios with a high variance of internet velocity. The lowest velocities might come from inter-urban areas where the user can search for a nearby charge station. Therefore it must be able to deal, at least its core features, with expected lower internet speeds.

3.7 Design Constrains

3.7.1 Standard Constrains

Firstly, all user data shall be treated in compliance with GDPR, or the equivalent privacy laws with jurisdiction in the country of use.

In second place, third part dependencies regulations must be followed.

Thirdly, the application should function on Android and IOS mobile systems.

3.7.2 Hardware limitations

As this system is design for applications in mobile devices (cellphones and/or car touchscreen systems) this hardware is the only requirement to run it. However, it must have Internet accessibility.

4 Formal Analysis

This sections aims user formal analysis to test rigorously the specification of the CPMS-eMPS system. To build the analysis, the tool used was Alloy: an open source language and analyzer for software modeling [1]. Moreover, the code was programmed with AlloyTools v5.1.0 [2].

The code was develop to describe the system attributes and constrains. Therefore, this sections describes its static properties and dynamics with Alloy signatures, facts and predicates.

The analysis was divided in four different models to simplify it. The first three are focus on the systems and the stakeholders relations, while the last one focus on describing some of their dynamics.

All models have the same basic structure with some differences to test several scopes. In other words, all contain the same mains signatures and facts, with personalize predicates and asserts to analyze its own cases.

The system has been modeled from the eMSP subsystem vision. Meaning, there is only one eMSP handling the user input, while there are multiples CPMSs, DSOs, CPOs and users.

Finally, to simply the analysis, it is considered that each CPO handles only one CS infrastructure. Therefore, the CPO model is a representation of the charging station and its operator all together. In that manner, the locations and the sockets are connected directly to the "CPO model".

4.1 Formal Model 1

The first model aims to picture the relations between the hardware and software subsystems. It contains the basic constrains that build the communications skeleton. The constrains and assumptions are list ahead:

- An eMSP can interact with multiple CPMSs and vice versa
- Each CPMS handles a single CPO
- Each CPO handles a group of sockets (can be only a unit)
- A CPO can interact with multiple DSOs and vice versa, managed by the CPMSs

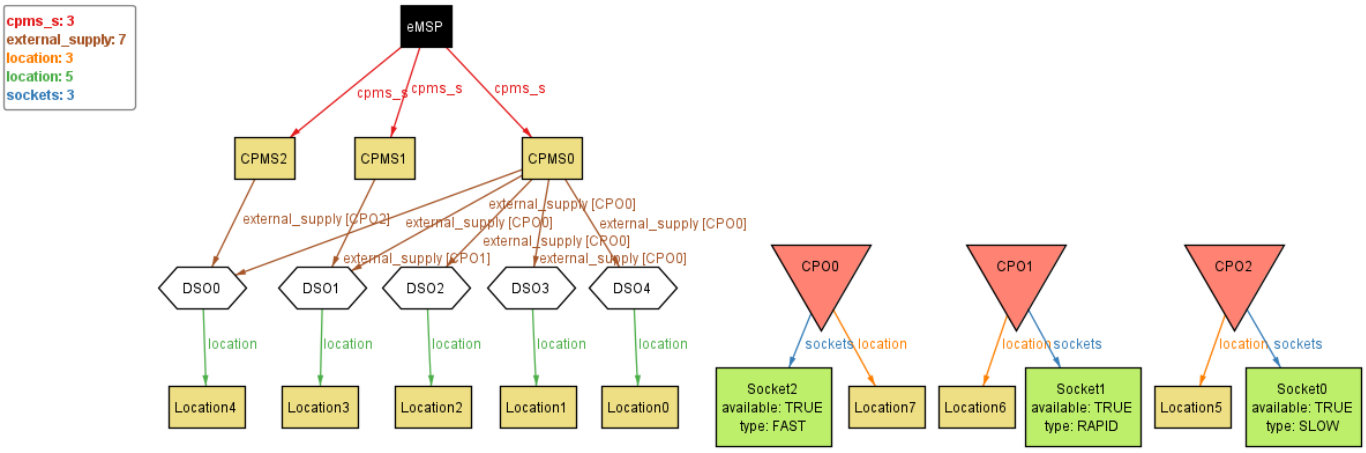

Figure 26: Example 1 Model 1

The examples shows how the eMSP, CPMSs, CPOs and DSOs conect. Its important to notice that all CPOs and DSOs must have different locations. Finally, the connections between the CPMSs and the DSO show which CPO is connected to which DSO.

4.2 Formal Model 2

This model aims to explore the user perspective and relations between bookings. To simply the data show in the graphic, the users information such as email, password and other data not relevant to this model is neglected. The constrains and assumptions adopted where the next:

- More than an user can be connected to the eMSP
- A socket becomes unavailable when it has been booked
- The booking requests are instantaneous: if a booking starts at time τ then it becomes unavailable for x $\geq \tau$
- There can not be any bookings of the same socket with overlapping time-frames

Finally, the clock figure represents the current time, like it has been taken an snapshot of the sockets and booking states.

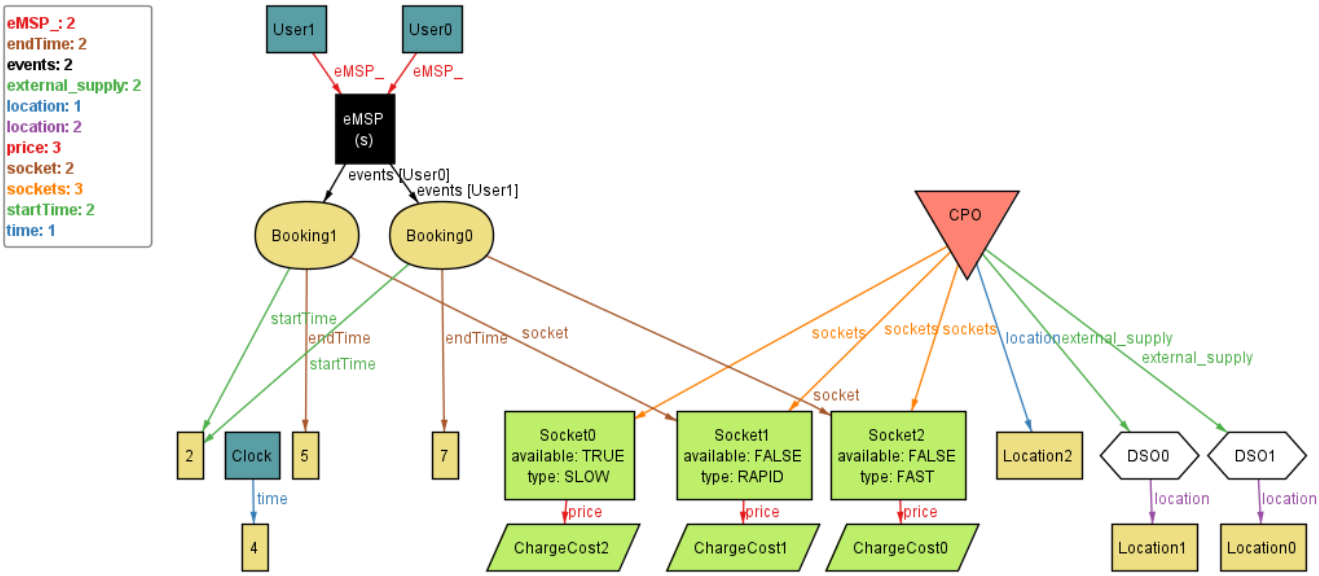

Figure 27: Example Model 2

In the image above there are shown 2 bookings and one vacant socket. The current time (represented by the clock) is $\tau = 4$. Both bookings start at $\tau = 2$ and finish after the current time. Therefore, their sockets are unavailable.

In the example, there are more bookings than sockets. The constrains imposed forbid two bookings with overlapping times to reserve the same socket. As an example, the bookings 1 and 0 have reserve the same socket:

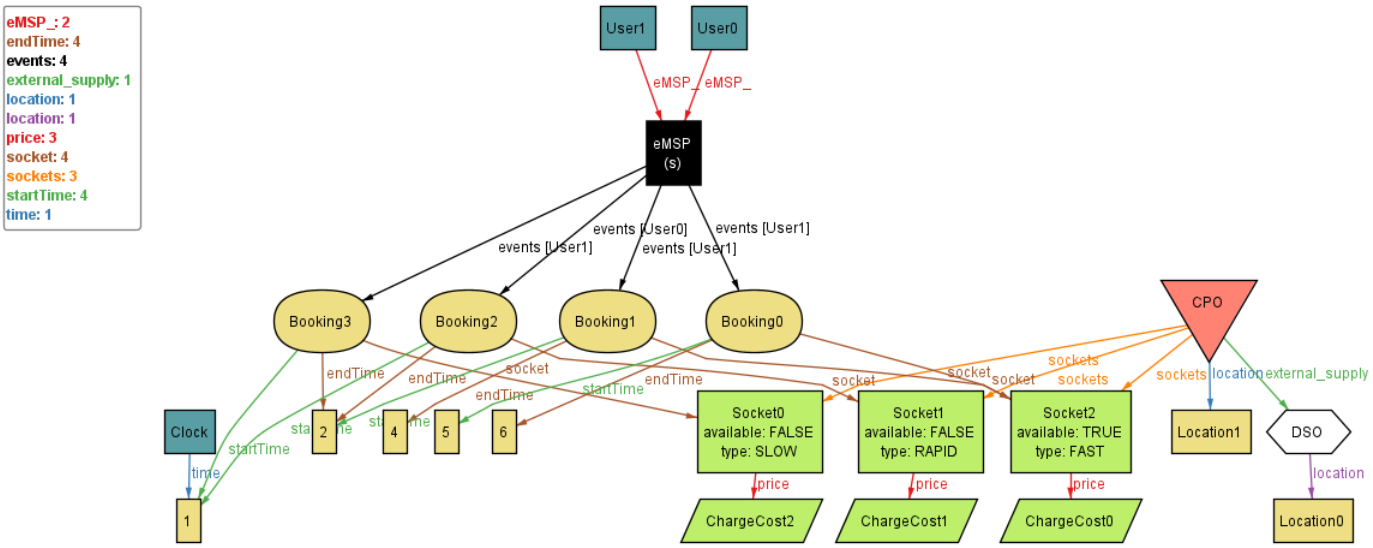

Figure 28: Example 2 Model 2

Again, only the sockets without a current booking are available for the current time $\tau = 1$.

Therefore, its clear how booking intervals don't over overlap, and how the sockets availability depend on their bookings.

4.3 Formal Model 3

This model aims to simulate the interaction between the DSOs energy prices and how the CPMSs manages the CPO energy supply. In that manner, the constrains and assumptions for this model are listed ahead:

- The CPMSs control the CPO energy supply: either the DSO, the internal batteries or a mix of both. This choice can be automated or manual.
- Not all CPOs must have internal batteries, however all must be connectec to at least one DSO
- All DSOs and CPOs must have a different location
- The CPMSs calculate the power supply cost per second of each socket depending only on its supply (each CPO can have a different supply cost) and its type of charge (slow/rapid/fast).

The next examples aim to describe several scenarios of power supply and charging service prices.

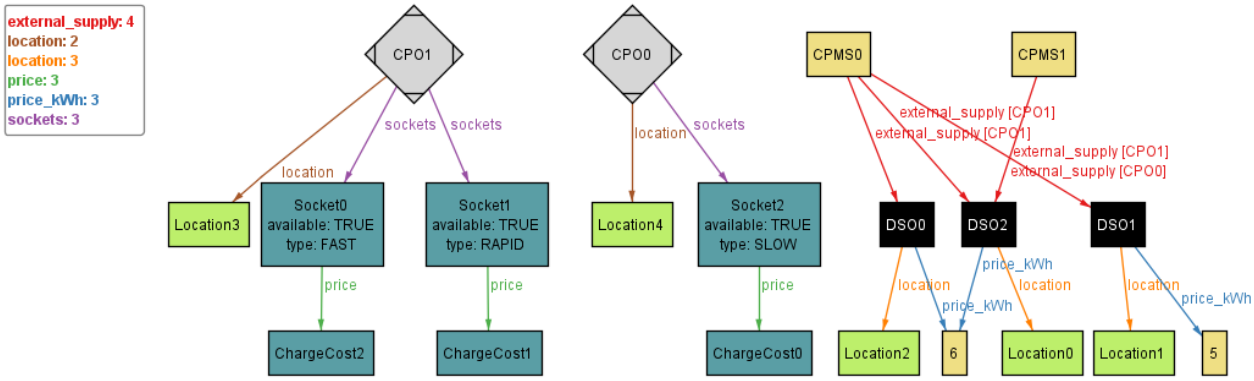

Figure 29: Example 1 Model 3

In the first example, none of the CPOs have internal supply, and each one is connected to a different set of CPOs (there can be intersections).

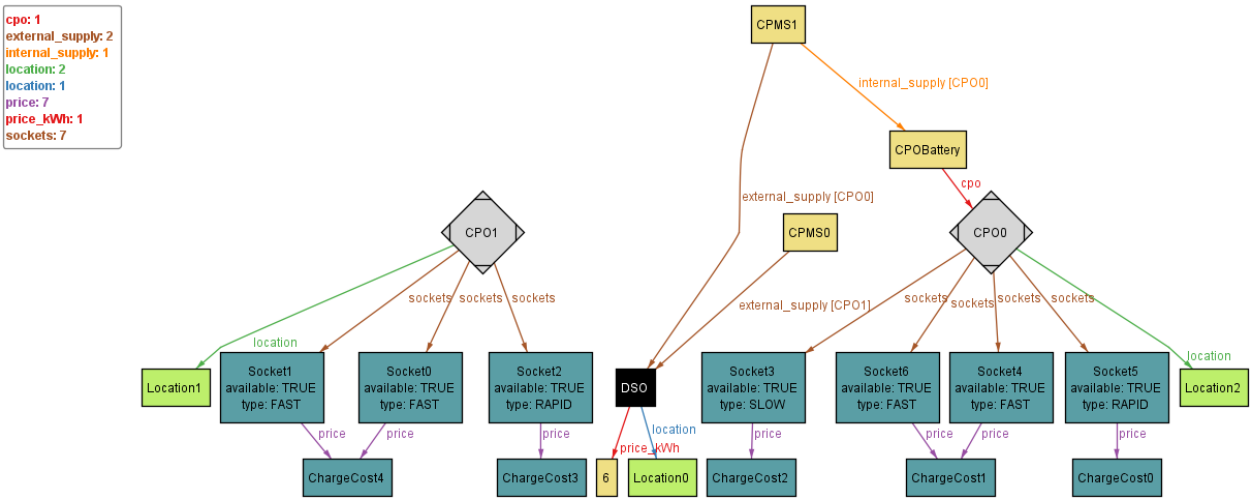

Figure 30: Example 2 Model 3

In the second example, only the second CPO has internal batteries. Again, its CPMSs manage their connections with the external supply, in this case, the same DSO. From the image, the *CPO1* has only external power supply and the *CPO0* has a mix of internal (batteries) and external (DSO) power supply. Moreover, it is important to notice how two sockets of the same CPO and type of charge, have the same charging price per second.

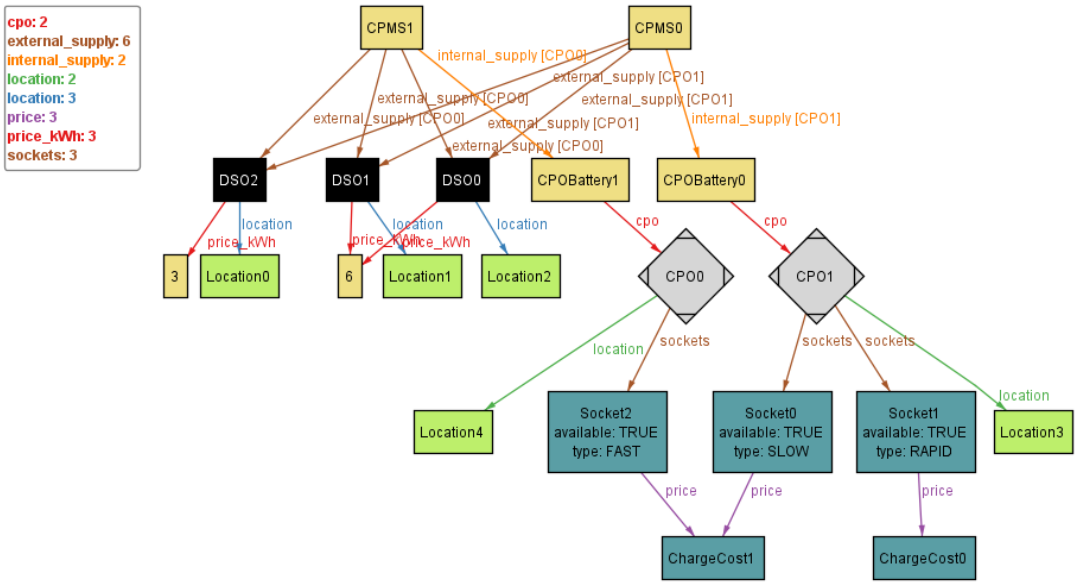

Figure 31: Example 3 Model 3

In the last example both CPOs have a mixed power supply: DSO + internal batteries. Moreover, both are connected to the same DSOs. It also shows how different CPOs can show the same price per kWh even in sockets of diferent type.

4.4 Static Models Code (1,2,3)

```
// HELP SIGNATURES - TYPES
sig ChargeCost {}
sig Location{}

abstract sig ChargeType {}
one sig FAST extends  ChargeType {}
one sig RAPID extends  ChargeType {}
one sig SLOW extends ChargeType {}

abstract sig Bool {}
one sig TRUE extends Bool {}
one sig FALSE extends Bool {}

// MAIN SIGNATURES - ENTITIES
sig CPOBattery { cpo: one CPO } // internal CPOs power supply: set of Batteries
sig Clock {  time: lone Int  }  { time > 0 }
```

```
sig User  {  eMSP_: one eMSP }

sig Booking {
    startTime: one Int,
    endTime: one Int,
    socket: one Socket,
} { startTime > 0
    endTime > startTime
}

sig Socket {
    id: one Int,
    available: one Bool,
    type: one ChargeType,
    price: one ChargeCost
} {  id>0
}

sig eMSP {
    cpms_s: some CPMS,
    events: User -> Booking
} { #(cpms_s) = #(CPMS) //all CPMs are connected to the eMSP
}

sig CPMS {
    external_supply: CPO -> DSO,
    internal_supply: CPO -> lone CPOBattery,
}

sig DSO {
    location: one Location,
    price_kWh: one Int
} {  price_kWh>0
}

sig CPO {
    location: one Location, //defined by ID
    sockets: some Socket,
}{ #(sockets) > 0
}

// HELP FUNCTIONS

fun get_cpo_of_cpms[cpms: CPMS]: set CPO {
 (cpms.external_supply).DSO
}
```

```
fun list_dsos_of_cpsm[cpms: CPMS]: set DSO {
 CPO.(cpms.external_supply)
}

fun get_dsos_of_cpo[cpo: CPO]: set DSO {
 cpo.(CPMS.external_supply)
}

fun get_bookings_of_user[u: User]: set Booking {
 u.(eMSP.events)
}

fun get_batteries[c: CPO]: set CPOBattery {
 c.(CPMS.internal_supply)
}

fun get_batteries[cpms: CPMS, cpo: CPO]: set CPOBattery {
 cpo.(cpms.internal_supply)
}


// FACTS

// *
// the constrains are divided by the structure they define.
// inside each one, there is a numerated list
// *


fact eMSPStructure{
 // 1: al/ bookings are connected to an user
 all b: Booking | #((eMSP.events).b) = 1
}

fact CPOStructure {

// *
// the scokets are considered part of the CPO structure
// *
    // 1: a socket can only belong to a CPO
    all disj c1, c2: CPO | #(c1.sockets & c2.sockets) = 0
    // 2: each CPO can have at max one battery (set of batteries)
    all disj b1,b2: CPOBattery | b1.cpo != b2.cpo
    // 3: all CPOs are connected to a DSO or more
    all cpo: CPO | #(get_dsos_of_cpo[cpo]) > 0
```

```
    // 4: sockets with the same type of the same CPO
    //    must have the same charging cost
    all disj s1, s2: Socket | all c: CPO |
        (s1 in c.sockets and s2 in c.sockets and s1.type = s2.type implies
            s1.price = s2.price )
        and
        (s1 in c.sockets and s2 in c.sockets and s1.type != s2.type implies
            s1.price != s2.price)

    // 5: a socket cannot have overlaping bookings
    all disj b1, b2: Booking |
        #(b1.socket & b2.socket) > 0
        implies (
        b1.startTime < b2.startTime and b1.endTime < b2.startTime )

        or (
        b2.startTime < b1.startTime and b2.endTime < b1.startTime )


    // 6: the socket must be unavailable while it is being used (booked)
    all b: Booking,c: Clock |
        (b.startTime <= c.time and b.endTime >= c.time)
        implies b.socket.available = FALSE
    all b: Booking,c: Clock |
        (b.startTime > c.time or b.endTime < c.time)
        implies b.socket.available = TRUE
    all s: Socket | #(Booking.socket & s)=0
        implies s.available=TRUE
}

fact CPMSstructure {
 all cpms: CPMS | (
 // 1: consistency constrain:
            the external(DSO) and interal(CPObattery)
            in a CPMS must be for the same CPO

 #(
     (cpms.external_supply).DSO+
     CPO.(cpms.internal_supply).cpo+
     (cpms.internal_supply).CPOBattery
 )=1 and

 // 2: all CPMSs are connected to at least one DSO
 (let dsos = list_dsos_of_cpsm[cpms] | #dsos > 0)  and
 // 3: CPMS can only detect its CPO batteries
```

```
 (all b: CPOBattery |
 b.cpo != get_cpo_of_cpms[cpms]
 implies #(get_batteries[cpms,b.cpo])=0
                else #(get_batteries[cpms,b.cpo])=1 )
 )
}

fact DSOstructure {
    // 1: all DSOs must be connected to a CPMS
    all dso: DSO | dso in CPO.(CPMS.external_supply)
}

// *
// General Constrains
// *

fact differentLocations {
    // 1: all DSOs and CPOs must have different locations
    no disjoint c1, c2: CPO |  c1.location = c2.location
    no disjoint d1, d2: DSO |  d1.location = d2.location
    all c: CPO, d: DSO | c.location != d.location
}

fact helpers {
    //all types must be associated to a socket
    #(ChargeType - Socket.type) = 0
    //all sockets must be connected to a CPO
    #(Socket - CPO.sockets) = 0
    // all user are connected to the eMSP application
    #(User - (eMSP.events).Booking) = 0
    // all locations must be associated to a DSO or CPO
    #(Location - DSO.location - CPO.location) = 0
    // all charge cost must be associated to a socket
    #(ChargeCost - Socket.price) = 0
    // all batteries must be from a CPO
    #(CPOBattery - CPO.(CPMS.internal_supply)) = 0
    // must be the same number of CPMS and CPOs
    #(CPMS) = #(CPO)
    // all sockets must have a diferent id
    all disj s1,s2: Socket,  cpo: CPO |
        s1.type = s2.type and
        #(cpo.sockets & s1 ) = #(cpo.sockets & s2 ) // and same CPO
        implies s1.id= s2.id else s1.id!=s2.id
}

// PREDICATED  - MODELS
```

```
//*
//Each model cuts some of the signatures to focus on its scope
//*

pred model1{
 #eMSP = 1
 #CPMS = 3
 #DSO = 5
 #CPO = 3
 #Booking = 0
 #User=0
 #Clock=0
}

pred model2{
 #eMSP = 1
 #Clock = 1
 #Booking < 5
 #DSO < 3
 #User > 1
 #CPOBattery = 0
}

pred model3 {
 #eMSP = 0
 #Booking = 0
 #User=0
 #Clock=0
 #CPO = 2
 #DSO < 4
}

//run model1 for 10
//run model2 for 15
//run model3 for 10
```

4.5 Formal Model 4

Finally this model aims to test the system dynamics. It describes the system response to different inputs. Not all inputs imply a system change, for example, trying to add an illegal booking (overlapping with other for the same socket) should not be allowed. The constrains are list ahead.

The next assumptions where tested with Alloy assertions **??**:

- The booking dynamics:

– A booking can only be added in a available socket (there is not overlapping between the new booking and the existing ones)
– A booking can be added and deleted (the system doesn't change)

```
// DYNAMIC MODELING

//*
// All from model 4, they are divied in scopes
//*

// 1:
pred addBooking[state, state': eMSP, b: Booking, u: User] {
    state'.events = state.events + u -> b
}

pred cancelBooking[state, state': eMSP, b: Booking, u: User] {
    state'.events = state.events - u -> b
}

assert noChangesInBookings {
    all state, state', state'': eMSP, b: Booking, u: User |
        no u.(state.events) and
        addBooking[state, state',b,u] and
        cancelBooking[state',state'',b,u]
        implies state.events = state''.events
}

// 2:
assert noOverLappingBookings {

    all state,state':eMSP,b1:User.(state.events),new_b:Booking |
        let existing_bookings = User.(state.events) |
        #(existing_bookings) > 1  and
        new_b not in existing_bookings
        and addBooking[state, state',new_b,User]
        // have the same socket
        and  #(b1.socket & new_b.socket) = 1
        implies (
            // must not no overlaping times
            (b1.startTime < new_b.startTime and
            b1.endTime < new_b.startTime )
            or
            (new_b.startTime < b1.startTime and
            new_b.endTime < b1.startTime )
        )
}
```

```
    //check noOverLappingBookings for 5
    //check noChangesInBookings for 15 but 5 Booking
```

- How sockets availability change over time: a socket becomes available when its booking as ended or been cancell

```
// 3:
pred timePasses[c, c': Clock] {
    c'.time > c.time
}

pred socketGetsAvailable[s, s': Socket] {
    // is the same socket, just the availability changes
    s.available = FALSE
    s'.id = s.id
    s'.type = s.type
    s'.price = s.price
    s'.available = TRUE
}

assert socketGetsFreed {
    all c, c': Clock, s, s': Socket, b: Booking |
        timePasses[c,c']
        and socketGetsAvailable[s,s']
        and s = b.socket
        implies (
            b.startTime <= c.time and b.endTime >= c.time
            and
            (b.endTime < c'.time or s' not in b.socket)
        )
}

check socketGetsFreed for 10
```

5 Effort spent

Task	Time spent
Team Discussion	9.5 h
Introduction	1.5 h
Overall Description	0.5 h
Specific Requirements	9 h
Formal Analysis	19 h
Total	39.5 h

Table 36: 10916443 - Claudia Baz Alvarez

Task	Time spent
Team Discussion	9.5 h
Introduction	2.5 h
Overall Description	6 h
Specific Requirements	22 h
Total	40 h

Table 37: 10832704 - David Alfredo Quintero Olaya

Task	Time spent
Team Discussion	9.5 h
Introduction	1.5 h
Overall Description	15 h
Specific Requirements	15 h
Total	41 h

Table 38: 10904511 - Pablo Llorente Cerezuela