

Grant number: 769016
Project duration: Sept 2018 - Aug 2021
Project Coordinator: Jacqueline Floch, SINTEF

HORIZON 2020: Mobility for Growth
MG-4.2-2017
Supporting Smart Electric Mobility in Cities
Project Type: Innovation Action



greencharge2020.eu

GreenCharge Project Deliverable: D5.3

Simulation and Visualisation Tools (revised version)

Editor: Antonio Esposito (SUN)

Authors:

Rocco Aversa, Beniamino Di Martino, Salvatore Venticinque, Dario Branco,
Antonio Esposito (SUN),

Svein Hallesteinsen (SINTEF), Geir Horn (UiO), Regina Enrich Sard (EURECAT)



www.civitas.eu

The research leading to these results has received funding from Horizon 2020, the European Union's Framework Programme for Research and Innovation (H2020) under grant agreement n° 769016

About GreenCharge

GreenCharge takes us a few important steps closer to achieving one of the dreams of modern cities: a zero-emission transport system based on electric vehicles running on green energy, with traffic jams and parking problems becoming things of the past. The project promotes:

<i>Power to the people!</i>	The GreenCharge dream can only be achieved if people feel confident that they can access charging infrastructure as and when they need it. So GreenCharge is developing a smart charging system that lets people book charging in advance, so that they can easily access the power they need.
<i>The delicate balance of power</i>	If lots of people try to charge their vehicles around the same time (e.g., on returning home from work), public electricity suppliers may struggle to cope with the peaks in demand. So we are developing software for automatic energy management in local areas to balance demand with available supplies. This balancing act combines public supplies and locally produced reusable energy, using local storage as a buffer and staggering the times at which vehicles get charged.
<i>Getting the financial incentives right</i>	Electric motors may make the wheels go round, but money makes the world go round. So we are devising and testing business models that encourage use of electric vehicles and sharing of energy resources, allowing all those involved to cooperate in an economically viable way.
<i>Showing how it works in practice</i>	GreenCharge is testing all of these innovations in practical trials in Barcelona, Bremen and Oslo. Together, these trials cover a wide variety of factors: <i>vehicle type</i> (scooters, cars, buses), <i>ownership model</i> (private, shared individual use, public transport), <i>charging locations</i> (private residences, workplaces, public spaces, transport hubs), <i>energy management</i> (using solar power, load balancing at one charging station or within a neighbourhood, battery swapping), and <i>charging support</i> (booking, priority charging).

To help cities and municipalities make the transition to zero emission/sustainable mobility, the project is producing three main sets of results: (1) *innovative business models*; (2) *technological support*; and (3) *guidelines* for cost efficient and successful deployment and operation of charging infrastructure for Electric Vehicles (EVs).

The *innovative business models* are inspired by ideas from the sharing economy, meaning they will show how to use and share the excess capacity of private renewable energy sources (RES), private charging facilities and the batteries of parked EVs in ways that benefit all involved, financially and otherwise.

The *technological support* will coordinate the power demand of charging with other local demand and local RES, leveraging load flexibility and storage capacity of local stationary batteries and parked EVs. It will also provide user friendly charge planning, booking and billing services for EV users. This will reduce the need for grid investments, address range/charge anxiety and enable sharing of already existing charging facilities for EV fleets.

The *guidelines* will integrate the experience from the trials and simulations and provide advice on localisation of charging points, grid investment reductions, and policy and public communication measures for accelerating uptake of electromobility.

For more information

Project Coordinator: Jacqueline Floch, Jacqueline.Floch@sintef.no

Dissemination Manager: Anne-Ingeborg Lund, anne-ingeborg.vanluijn@pnoconsultants.com

Executive Summary

This document focuses on the description of the tools developed starting from the prototypes initially provided in deliverable D5.2. Initial requirements, design and prototypes have been updated and extended, and use cases have been reported to support users in approaching the developed tools.

In particular the deliverable starts with highlighting the role of simulation in the GreenCharge framework, and the importance of providing instruments to calculate and graphically visualize KPIs, in relation to evaluation activities carried out within the GreenCharge project. These tools acquire even more importance if we consider the possibility to simulate scenarios involving larger and more diverse energy smart neighbourhoods than realized in the pilots, based partly on data collected by the pilots and partly on data collected in other contexts.

Models used to analytically represent different appliances, including energy consuming, producing and storing devices, are provided in this deliverable, explaining how they are used to configure and customize the simulation facility.

The overall architecture of the GreenCharge Simulator is provided, together with a description of its components. Particular emphasis is put on optimisers' interfaces and approaches.

Interaction between the GreenCharge Simulator and the GreenCharge Energy Management Systems (EMS) is provided through an interaction protocol, completely described in terms of exchanged messages together with usage examples.

The deliverable presents the Simulation and KPI Calculator and Visualization tools by focusing on both the offered functionalities and provided GUIs.

Graphical Interfaces are indeed a relevant part of the software description, as they provide a facilitated point of access for users and are thus described in details to provide a quick reference user guide. However, pointers to online guides are also provided, so that users have a complete reference manual at their disposal.

The document explains how the simulation tool can be deployed, configured, and customized, by following a use case scenario already presented in D5.2.

Since the current Deliverable is focused on the development of the Simulation Tool, of the KPI Calculation and Visualization tools, and of the Optimisers, no further details are provided regarding simulated tasks and models, unless necessary to describe the design of the aforementioned tools and their components.

Table of Contents

Executive Summary	1
List of Abbreviations	6
1. About this Deliverable	7
1.1. Why would I want to read this deliverable?	7
1.2. Intended readership/users.	7
1.3. Other project deliverables that may be of interest	7
2. Role of the Simulation in the evaluation activities of the project	8
2.1. Assessment of objectives and key performance indicators.....	8
2.2. GreenCharge scenarios and simulation requirements	9
2.3. Definition of simulation scenarios	11
3. Modelling and Configuration of the simulation environment.....	13
3.1. Modelling energy demand, production and storage	13
3.1.1. PV panels	13
3.1.2. Shiftable devices (appliances)	13
3.1.3. Heaters/coolers	14
3.1.4. Charging EVs and Stationary batteries	15
3.1.5. Background loads	16
3.2. Configuration of the simulation environment with the involved energy actors	16
3.2.1. Neighbourhood Configuration	18
3.2.2. Loads Configuration.....	20
3.3. Graphic User Interface.....	21
3.3.1. Settings	21
3.3.2. Control Panel.....	21
3.3.3. Show Results.....	22
3.3.4. GreenCharge Simulation Tool Info	22
3.4. The Configuration phase using the tool GUI.....	22
3.4.1. Configuration of the Neighbourhood	23
3.4.2. Definition of the Neighbourhood Loads	24
3.5. Configure and run a simulation associated with a specific scenario	25
4. GreenCharge Simulator: Software Architecture and Components.....	26
4.1. Overview of the GreenCharge Simulator Software Architecture	26
4.2. GreenCharge Simulator Components	27
4.3. EMS Interaction Protocol.....	29
4.4. UiO distributed optimiser	47

4.4.1.	Optimisation approach.....	47
4.4.2.	Optimiser Interface	48
4.4.3.	The Energy Group.....	49
4.4.4.	Consumers.....	50
4.4.5.	Producers.....	51
4.4.6.	Prosumers.....	52
4.5.	EURECAT Optimiser	53
4.5.1.	Optimiser approach.....	53
4.5.2.	Optimiser Interface	53
4.5.3.	Optimiser engine	55
4.5.4.	Loads	55
4.5.5.	Electric Vehicles.....	56
4.5.6.	Generators.....	57
4.5.7.	Grids	57
4.5.8.	Batteries	57
4.5.9.	Neighbourhood	57
5.	The KPI Calculator and Visualization Tools.....	58
5.1.	The GreenCharge KPI Calculator.....	58
5.2.	Overall Design of the Calculator	59
5.3.	Data Ingestion and Curation	60
5.4.	The Visualization tool	61
5.5.	Software Technology Used	67
6.	Deployment and utilization of the open source simulator.....	68
6.1.	Design and use of the Container Based Deployment Configuration	68
6.1.1.	The open source GitHub repository	68
6.1.2.	How to build the simulation platform	69
6.2.	Running the Simulator	69
6.2.1.	For the Developer.....	70
6.2.2.	For the Evaluator	71
6.3.	Visualization of Results	73
6.3.1.	Report Generation.....	73
6.4.	Creation of a new Simulation Scenario.....	77
6.5.	Example of Test Scenario.....	78
6.5.1.	Configuration files	80
7.	Conclusions	86
8.	External Links to Tools and Guidelines.....	87

Table of Figures

Figure 1 GreenCharge Evaluation Loop.....	9
Figure 2 Flexibility of heating/cooling device. The horizontal axis represents time and the vertical axis represents cumulated energy.....	14
Figure 3 Flexibility of Charging EV.....	16
Figure 4 GreenCharge Control Panel	21
Figure 5 Simulator Dashboard.....	22
Figure 6 Setting Use Case	23
Figure 7 Create Neighbourhood Use Case a).....	24
Figure 8 Create Neighbourhood Use Case b).....	24
Figure 9 Insert Devices Loads Use Case	25
Figure 10 GreenCharge Simulator Architecture Overview	26
Figure 11 GreenCharge Files and Directory Tree	28
Figure 12 Simulation Scenario: configuration of the neighbourhood.	30
Figure 13 Main dashboard of the integration tool.	42
Figure 14 Simulation dashboard.....	43
Figure 15 XMPP implementation of the dummy EMS.	44
Figure 16 Rest implementation of the dummy optimizer.....	45
Figure 17 RAW visualization of the REST protocol.....	45
Figure 18 Tree view visualization of completed simulations.....	46
Figure 19 Visualization of simulation output.	47
Figure 20 List of exchange messages.	47
Figure 21 The interface class of the optimiser has one message handler with a corresponding message class for each of the types of messages that can be received from the Simulator's event dispatcher. The message classes are derived from common base classes supporting serialisation to and from the right XMPP message format.	49
Figure 22 The energy group receives messages to create and manage categorised <i>energy objects</i> , and grid related parameters like the energy cost profile and the types of energy offered from the grid over time.....	50
Figure 23 The consumer types of the distributed optimisers.....	51
Figure 24 The producer types of the distributed scheduler	51
Figure 25 The <i>prosumers</i> that are able to act as both a consumer and a producer of energy in a smart neighbourhood as implemented in the distributed optimiser.....	52
Figure 26 Mapping between the optimiser and the simulator objects.....	54
Figure 27 Internal representation of shiftable loads in Eurecat optimiser.....	56
Figure 28 Internal representation of electric vehicles in Eurecat optimiser	56
Figure 29 Internal representation of a battery.....	57
Figure 30 GC-Calculator Scheme.....	59

Figure 31 DataChecker General View.....	61
Figure 32 DataChecker Detailed View.....	61
Figure 33 Login Page	62
Figure 34 Visualization Tool Top View.....	63
Figure 35 Visualization Tool Legend Help Bar	63
Figure 36 Evaluation History Table	64
Figure 37 Date Selection	64
Figure 38 Loading Screen.....	65
Figure 39 Example of Visualization of Some KPIs.....	65
Figure 40 Self Consumption Chart.....	66
Figure 41 Power Peak KPI	66
Figure 42 Business KPI example.....	67
Figure 43 Container based deployment configuration.....	68
Figure 44 Folder tree of a simulation scenario.	70
Figure 45 Web access to the GUI container.	72
Figure 46 Control Tab	73
Figure 47 Synthetic parameters of the simulation output.....	73
Figure 48 Coherence checking results.	74
Figure 49 Visualization of times-series.	76
Figure 50 Energy related KPI computation on simulation output.....	76

List of Tables

Table 1: List of abbreviations.....	6
Table 2 Energy Management related Measures implemented in GC pilots	10
Table 3 Monitored and controlled devices in the GC pilots.....	11
Table 4 Varying characteristics	12

List of Abbreviations

Table 1: List of abbreviations

Abbreviation	Explanation
V2G	Vehicle to grid
CP	Charge point
GC	GreenCharge
EV	Electric Vehicle
KPI	Key Performance Indicator
EMS	Energy Management System
CMS	Charge Management System
NEMS	Neighbourhood Energy Management System
JID	Jabber identifier
DSO	Distribution System Operator
TSO	Transmission System Operator

1. About this Deliverable

1.1. Why would I want to read this deliverable?

The content of this deliverable focuses on the design and development of software tools used for simulation and visualization, which will complement the other evaluation activities and tools envisaged in the project. This document extends the content provided by deliverable D5.2, providing a more in-depth description of the requirements, design, software architecture and implementation of the software. The extended description of usage scenarios and the illustration of the intuitive developed GUI serves as a quick user guide for using the tool, while external references to full utilization guides have also been provided.

1.2. Intended readership/users.

This deliverable is essentially aimed at project partners who are involved in the evaluation activities and want to use simulation, KPI calculation and visualization tools to complement the analysis of the data recorded in the pilots. This document and the available software made may also be of interest to users who intend to deepen their understanding of the conceptual model underlying the innovative technologies introduced by the GreenCharge project.

1.3. Other project deliverables that may be of interest

- *D5.1&D6.1 Evaluation Design / Stakeholder Acceptance Evaluation Methodology and Plan for the design of the GreenCharge evaluation methodology.*
- *Deliverable 4.1: Initial Architecture Design and Interoperability Specification to frame the simulation tool into the GreenCharge reference architecture for smart and green charging.*
- *Deliverable D2.1: Initial Strategic Plan for Pilots to identify and design the possible simulation scenarios inspired by the use cases developed in the Pilots.*
- *Deliverable D5.2: Simulation and Visualisation Tools (initial version) to understand the preliminary requirements and design specification on which the current simulation and visualization tools have been developed.*
- *Deliverable D5.4: Intermediate Evaluation Results to understand the relationship between indicators defined in the evaluation framework.*

2. Role of the Simulation in the evaluation activities of the project

As described in Deliverable D5.1/D6.1 one of the main objectives of the GreenCharge project is the evaluation of the impact on e-mobility that the specific technology can provide when it is used in a certain pilot.

In general, three evaluation methods will be used in the GreenCharge project.

- *Evaluation of stakeholder acceptance based on manual data collection methods, such as surveys and interviews.* Workshops will be organized in each Pilot. Surveys will be distributed to participants and to volunteers who are available to provide their feedbacks manually or using on-line tools
- *Evaluation of technology with analysis based on automatic data collection.* In GreenCharge Pilots user's behaviour and energy utilization in charging stations will be monitored. Data will be collected automatically, e.g., via system logs, user Apps, while different business models and supporting innovative technological solution are applied. Data analytics and numerical models will be used to evaluate KPIs.
- Evaluation of technology based on simulations.

The use of simulation for evaluation purposes arises from the need to overcome the intrinsic limitations of Pilots in terms of scalability, configuration, data availability, regulation, and time and effort constraints. The objective of this document is to identify the software requirements and the kind of scenarios to simulate potential limitations of the simulation approach. Furthermore, the importance of the role of the simulation in the project evaluation activities is summarised here for the convenience of the reader.

2.1. Assessment of objectives and key performance indicators

The GreenCharge evaluation methodology will be based on CIVITAS Evaluation Framework. CIVITAS is a network of cities dedicated to cleaner, better transport in Europe and beyond. Since it was launched by the European Commission in 2002, the CIVITAS Initiative has tested and implemented over 800 measures and urban transport solutions as part of demonstration projects in more than 80 Living Lab cities Europe-wide.

GreenCharge has adopted and customized the CIVITAS Evaluation Framework, focusing on e-mobility, in order to exploit the valuable results of that project in terms of methodology and procedures, but also to contribute to the work of the CIVITAS network.

The main concepts of CIVITAS Evaluation Framework are *Measures* and *Key Performance Indicators (KPIs)*. A *Measure* defines one or more aggregated activities which are implemented to have a positive impact on city transport. A *KPI* corresponds to one or more aggregated parameters which allow for a quantitative evaluation of the impact produced by a Measure on a specific aspect of transport. The adoption of the evaluation framework has consisted in the specialization of Measures and KPIs related to the e-mobility context, with a special attention to technology evaluation and to the assessment of stakeholder acceptance. As it is shown in the evaluation loop of Figure 1, aggregated technologies and business models will be operated by Pilots and evaluated in different WPs. Data will be collected at different milestones of the project. Intermediate and final KPIs evaluations will be scheduled. Evaluation of results consists of a quantitative and a qualitative analysis for a subset of defined KPIs, which will be selected in different Pilots according to their relevance in the specific context. As it is shown in **Figure 1**, when feasible, the evaluation of KPIs should highlight comparison to baseline values, for better identifying the impact related to the innovation introduced by the GreenCharge technology.

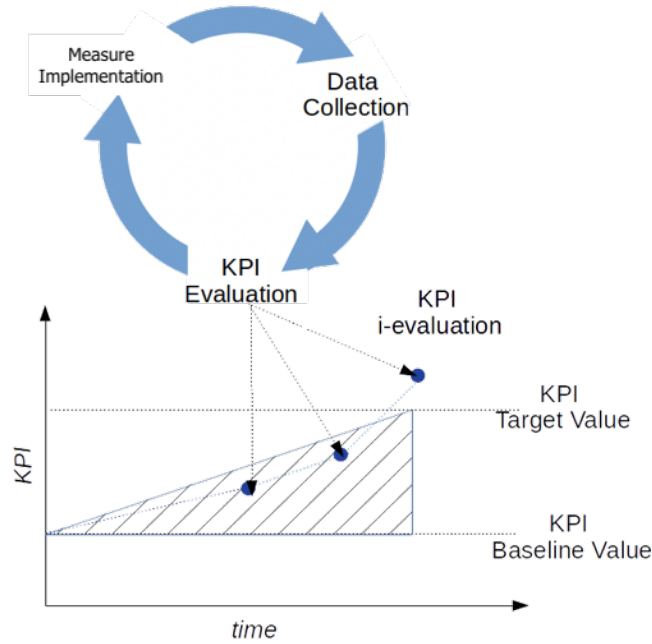


Figure 1 GreenCharge Evaluation Loop

Simulation will be used in the *GreenCharge Evaluation Loop* to operate the measure in a virtual environment where the Pilots can be extended overcoming real limitations and the measure can be easily complemented with missing functionalities.

This kind of approach will also allow the evaluation of KPIs at design time, to predict the return of investment or to optimize the dimensioning and positioning of new charging stations, as well as the acquisition of new EVs.

2.2. GreenCharge scenarios and simulation requirements

Analysis of the data collected in the pilots alone only allows the evaluation of the implemented instances of the measures. Due to regulatory and budgetary constraints, and the limited duration of the project, these are rather few, implemented in small scale, and not necessarily representative of a future with much higher density of EVs and a more ubiquitous and smarter charging infrastructure and energy supply system than we see today. Therefore, in order to broaden the basis for the evaluation we apply simulation based on data collected in the pilots.

With simulation we can simulate the impact of the GC concept in a more diverse set of scenarios, both with respect to size of the ESNs and diversity and dimensioning of included measures. We focus on the impact of measures related to smart charging and smart energy management in neighbourhoods. The selected measures are listed in Table 2.

The output from a simulation is an optimised consumption profile for all the consumer devices involved, and for the batteries both the charging and discharging profile.

Table 2 Energy Management related Measures implemented in GC pilots

Id	Class	Description	Implemented in	Corresponding D5.4 sub-measure(s)
1	Flexible charging	Drop-in charging with initial and target SoC and planned departure time provided at arrival	OSL D1	Flexible charging
2	Booked charging	Charging is booked in advance, planned arrival and departure time and initial and target SoC provided at booking time	OSL D2, BRE, BCN	Booking of charge point + reservation of energy for charging
3	Priority charging	If energy availability constraints prevent satisfaction of the charging constraints of all connected EVs, the CMS will seek to distribute available energy among connected EVs in a fair manner. Users may pay extra to take priority in this process. May be chosen at booking or arrival time.	OSL, BRE	Priority charging
4	Energy smart neighbourhood (ESN)	A local Energy management system seeks to shift and modulate flexible loads and control the behaviour of local storage in a building or neighbourhood in an optimal way according to given optimization criteria and constraints.	OSL, BCN, BRE	Optimal and coordinated use of energy
5	Local RES	Local renewable electric energy production, e.g., solar panels	OSL, BRE, BCN	Use of local RES
6	Local storage	Local electric energy storage, e.g., stationary battery.	OSL, BRE, BCN	Use of stationary energy storage
7	V2G ¹	Ability to exploit discharging of EVs connected for charging when possible within constraints set by user and beneficial for optimal demand profile of the building or neighbourhood.		Exploiting V2G

¹ Requires EVs and CPs supporting discharging and an EMS able to exploit it. None of the pilots include EVs and CPs supporting V2G, so it's impact can only be investigated in simulations.

9	Shared EVs	The fraction of EVs that are available for shared use	BRE	
---	------------	---	-----	--

An overview of the devices involved in the implementation of the measures implemented in the pilots is provided in Table 3. For the simulations we need to simulate the energy demanding and/or supplying behaviour of these devices, based on relevant data from their technical description and on logs of their behaviour collected during the operation of the pilot installations.

Due to the limited time available for collecting data, we will not have time to collect both baseline data and data from the situation with measures in place in all demos. However, for the purpose of the selected measures, we can estimate a reasonable baseline by simulating with an objective function minimising the delay. Then all demands are satisfied as early as possible while respecting capacity constraints in the local distribution grid, which is the behaviour one may expect with modern EV charging equipment. The simulation facility must support this.

Table 3 Monitored and controlled devices in the GC pilots

Kind of device	Number (approximately)	Description	Control
EV charge point	50	Max charging power ranging from 3,6 to 22 kW, both private and public use	power modulation
EVs charging at the charge points	50	Battery capacity varying from 15 to 100 kWh	n/a
Water heater	4	Large ones heating	on/off
Room heating	8	Electrical heating devices inflats	on/off
eBike charger	5	At bike sharing facility	power modulation
eScooter battery charging hub	1	Part of battery swapping service for rental eScooters	power modulation
Stationary battery	3	Capacity varying from 1.3 to 50kWh	charge/discharge
Solar plant	4		n/a

2.3. Definition of simulation scenarios

Based on the elements described in section 2.1 we have proposed 9 base scenarios. Three of them correspond to the implemented demonstrators and may provide interesting feedback to demo owners about how the installations will behave in possible future scenarios. The remaining 6 are artificial scenarios created by combining and/or replicating elements from the demonstrators and representing ESNs of varying size, complexity, and context, closer to the project vision than the implemented ones, and thus allowing to investigate the impact of more full-fledged deployment of the GreenCharge concept.

For each base scenario we will run a number of simulations varying systematically one characteristic of the scenario at a time and computing the indicators. In this way we will investigate to which extent and in which way the varying characteristics impact the relevant indicators. The varying characteristics are listed in Table 4. Mostly they correspond to the presence and dimensioning of the measures implemented in the

demonstrators. The dimensioning in some cases corresponds directly to indicators of the evaluation framework (see D5.4).

The size and diversity of the base scenarios vary from a small solar powered charge station (a few charge points with a solar panel and stationary battery), to a full-fledged ESN with several hundred households, with both private and public charge stations, electric heating and cooling devices and a significant amount of local energy production and storage capacity as well as a realistic amount of non-flexible background loads. The scenarios are documented in more detail in D5.4.

Table 4 Varying characteristics

Varying characteristic	Variation
Local energy management	Optimisation method (centralised or distributed) and optimisation criteria (minimise energy cost or maximise energy greenness)
Number of EVs	EV penetration (e.g., 25%, 50%, ... 100%)
Local RES	Percentage of total consumption, e.g., 0.25, 0.5, ...
Stationary battery storage	Capacity as % of total consumption%
Grid connection capacity	% of average consumption
Internal transfer capacity	With and without constraints, gradual removal of bottlenecks
Price model for the calculation and sharing of energy cost	E.g., energy only or mixed energy/power, fixed or Time of Use (ToU) or spot
Share of booking	Depends on available data
Share of battery capacity for V2G	Select EVs arbitrarily. Share of capacity drawn from distribution.

3. Modelling and Configuration of the simulation environment

In this section, we describe the simulation environment related to a general GreenCharge scenario where the Neighbourhood energy management system configured in the simulator should take in account the additional power requests coming from the nearby Charge management system.

The GreenCharge Simulator can be found at:

<https://github.com/GreenCharge/gcsimulator>

Guidelines related to the Simulator have been published at:

<http://parsec2.unicampania.it/~branco/gcccalculator/simulatorUserGuide.pdf>

3.1. Modelling energy demand, production and storage

In this section, we describe in detail the modelling of the different involved energy actors with their respective roles:

- Producers, i.e., electric energy producing devices,
- Consumers, i.e., electric energy consuming devices
- Prosumers: i.e., electric energy storing devices that switches between acting as consumers (when charging) and as producers (when discharging).

As can be seen from the overview of the monitored and controlled devices of the pilot sites in Table 3, we have one kind of pure Producer: PV panels; 3 kinds of Consumers: appliances like washing machines and dishwashers and the like, heating and cooling devices and EVs connected to a CP for charging; and one kind of prosumers: stationary batteries. In addition, although not present in any of the pilot sites, we also include V2G enabled charging EVs in order to investigate the potential impact of this technology, which is expected to become commonplace in the future. Finally, we have a number of consumers with a non-flexible demand that need to be taken into account. Their aggregated load is estimated from historical data and treated as an additional consumer (background load).

We simulate the energy related behaviour of these devices as a sequence of load and production events with given flexibility. These are input to an optimising algorithm that computes an optimal production and/or consumption profile for each device. The modelling of flexibility could be seen as specifying a corridor for the load profile, plus some constraints on the shape (as described in detail in the next subsections).

3.1.1. PV panels

The energy production from PV panels varies over time depending on the insolation, the orientation of the panel surface relative to the position of the sun on the sky, ambient temperature and the degree of shadowing from neighbouring objects. In real time energy management, the PV production has to be predicted based on weather forecasts for some time into the future, and is updated when updated weather forecasts become available, usually before the timespan of the previous prediction has ended. Then it is always the latest prediction that is valid. The prediction is modelled as a time series (sequence of time and produced energy pairs) where each element gives the cumulated energy produced from the beginning of the time series until the given time.

3.1.2. Shiftable devices (appliances)

These are devices like dishwashers and washing machines that once started have a fixed consumption profile, but that are flexible with respect to when they start within a given time window. They are modelled by a timeseries representing the fixed consumption profile, and the time window during which this demand must be satisfied, given as the earliest start time and the deadline for finishing.

3.1.3. Heaters/coolers

Nowadays, in most cases these devices are controlled by a thermostat turning them on and off to maintain a given temperature in the heated or cooled space. The actual constraint on such loads is the setpoint temperature and accepted deviation and we can shift and vary the length of the on periods as long as we keep the temperature inside the given bounds. In order to use that directly we would need a thermodynamic model of the house, enabling us to simulate the temperature variation in a heated or cooled space caused by a particular consumption profile.

The construction of such models is being studied in another project in SINTEF and it appears to be fairly complicated. Also, there are energy management products for households on the market claiming to “learn the house”, using machine learning techniques. We adopt a simple approach in this spirit and represent the flexibility of an h/c device as a “corridor” for the accumulated energy, constructed based on the recorded energy consumption as illustrated in Figure 2.

This corridor represents a narrow energy band that approximates the consumption profile of these device. Any consumption profile with the stairs-like shape dictated by the on/off nature of the device control and the power demanded when on, and fitting inside the corridor, would be acceptable. The width of the corridor reflects the allowable deviation from the setpoint temperature. It could vary during the day: narrower in periods with comfort temperature selected and wider otherwise.

In practice this kind of devices tend to be turned on or off for long periods. Thus, the consumption period can be much longer than the planning horizon of the scheduler, but for convenience we split it into 24 hrs long loads midnight to midnight. It is represented as two time series, one for cumulated energy and one for the allowed deviation (i.e., the width of the corridor).

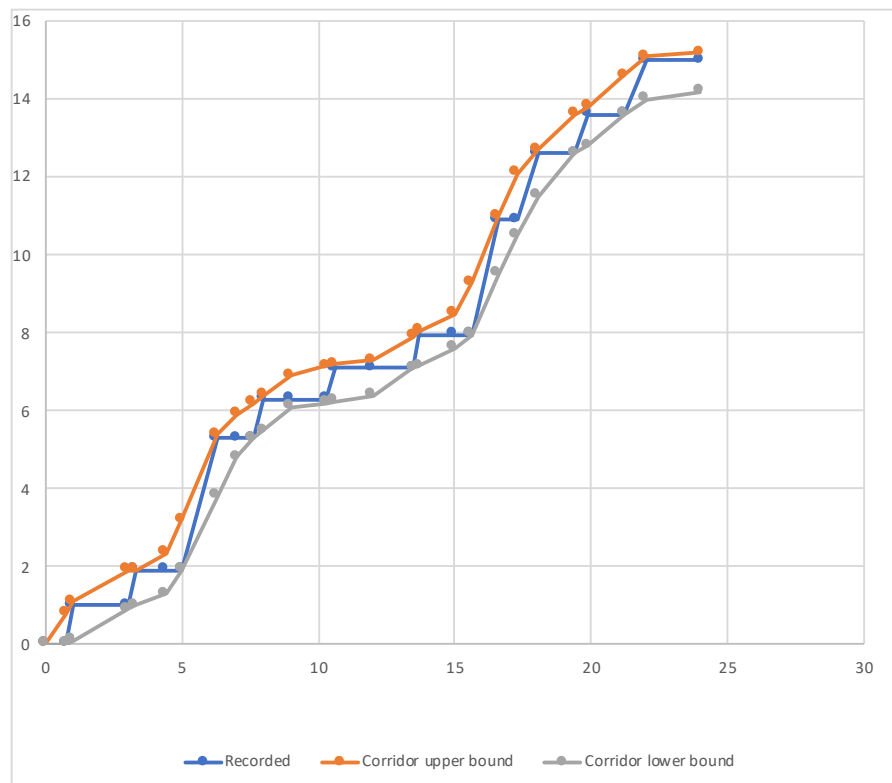


Figure 2 Flexibility of heating/cooling device. The horizontal axis represents time and the vertical axis represents cumulated energy

3.1.4. Charging EVs and Stationary batteries

A charging EVs connected to a modern CP enable both shifting and shaping of the power demand, within constraints as illustrated in Figure 3. The constraints are dictated partly by user needs and partly by technical parameters of the EV battery and battery management system and/or the charger.

The constraints dictated by the user are:

- *Arrival time*, i.e., the time when the EV arrives and connects to the CP. In the case of a booked charging, it will be a planned arrival time until the EV arrives, and then it may be adjusted to reflect the actual arrival time.
- *Planned departure time*, i.e., the time when the user expects to need the car again.
- *The target SoC [%]*, i.e., the SoC needed at departure. Here it is assumed the target SoC is understood as min SoC at departure, meaning that any SoC higher than that would also be ok. SoC is given as a percentage of a fully charged battery.
- *Min SoC for V2G [%]*. i.e the lowest SOC allowed in cases where V2G is enabled.

The constraints dictated by the EV battery are (unit in square brackets)

- *Battery capacity [kWh]*, i.e., the amount of energy that can be stored in the battery
- *Initial SoC [%]*, i.e., the state of charge at entry
- *Max charging power and max discharging power [kW]*. We adopt a linear model where the max (dis)charging power is constant until a given SoC threshold and then decreases linearly to 0 when the battery is full (when charging) or empty (when discharging). This is not a completely accurate model but is considered a sufficiently accurate approximation for our purpose.
- *Threshold above which the max charging power is decreasing [%]*.
- *Threshold below which the max discharging power is decreasing [%]*.
- *Efficiency*, i.e., the fraction of the energy charged into the battery that can be retrieved by discharging.
- *Cost of use [€/kWh]*. reflects that the capacity of the battery deteriorates with the number of charging/discharging cycles.

The battery management system protects the battery against excessive wear and risk of damage by preventing discharging it completely and also charging it fully. We model only the usable capacity.

We use the same model for EVs and stationary batteries. The flexibility is indicated by the shaded area in Figure 3. Any demand profile that stays within the shaded area is acceptable, with the exception that for a charging EV without V2G enabled, there is the additional constraint that charging is not possible (the cumulated power must be increasing monotonously).

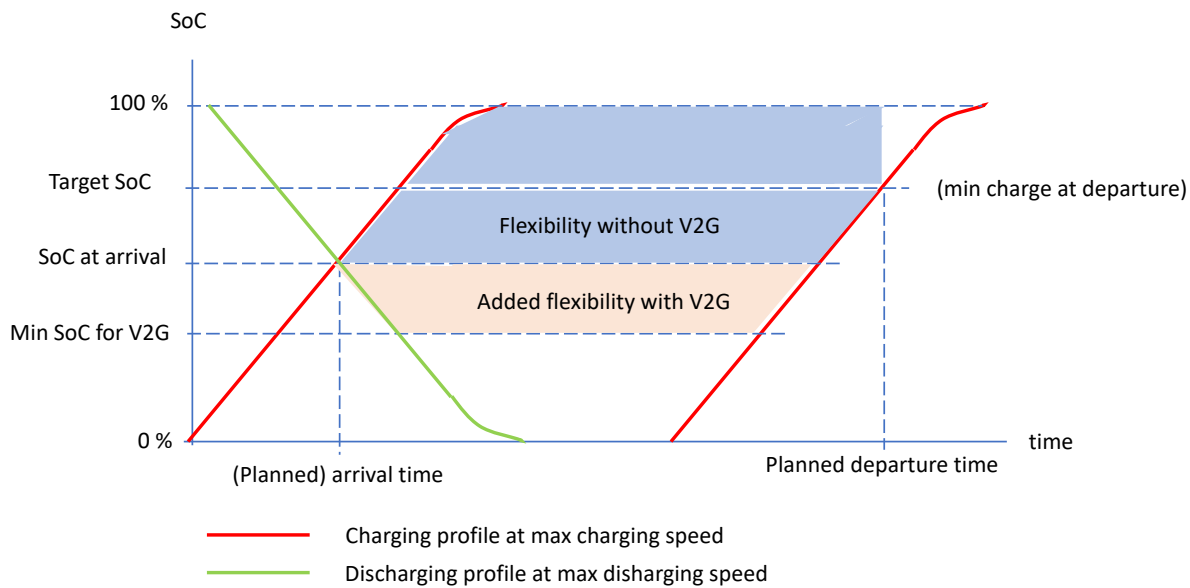


Figure 3 Flexibility of Charging EV

3.1.5. Background loads

By background load we mean that part of energy consumed in a house or building that cannot be controlled or scheduled as individual consuming devices but that must be taken into account in the optimization phase. In a real scheduler we do not know the background load in advance, so we have to predict it based on historical data. In a simulation based on recorded data we have available the recorded background load for the simulation period. The description of a simulation session should contain both the predicted and actual background load in order to enable analysis of the effect of different approaches to predicting background load and prediction accuracy.

3.2. Configuration of the simulation environment with the involved energy actors

In this section, we describe, as first scenario of use to test the initial version of the prototype, a simplified simulation environment. In this scenario are included the first three Use Cases foreseen in the Oslo pilot, obviously with increasing levels of complexity. In the following we summarize for each of the Oslo Use Cases a short description of the scenario together with the related challenges when implementing them in the pilot:

Oslo UC#1 Normal charging in the garage

Description:

- Individually owned parking places – typically overnight charging
- App for input of next time of use (and state-of-charge)
- System schedules charging based on other electricity consumers (other EVs, heated driveway) and on availability of solar power (on rooftop) and state of charge of stationary battery

Main Challenges for pilot implementation:

- User needs to buy own charging point + establishment of business agreement
- User willing to provide accurate information to **Charge Management System (CMS)**
- Integration of CMS and **Neighbourhood Energy Management System (NEMS)**

- Procurement of solar power and stationary battery

Oslo UC#2 Long-term parking (V2G possibilities) in the garage

Description:

- Extension of UC#1 for EVs parked for longer

Main Challenges for pilot implementation:

- Availability of vehicles with V2G
- Users willing to provide their in-vehicle battery (rewards)
- Users willing to provide necessary input to the system (time of departure)

Oslo UC#3 Drop-in charging

Description:

- Making existing outdoor chargers available to external visitors or employees at nearby school
- 4 semi-fast chargers – currently used by residents through simple booking system (Google docs spreadsheet)
- Access will be through app

Main Challenges for pilot implementation:

- Inform and attract potential users – sufficient usage required to ensure return of investment
- Establishment of business agreement
- Motivate download of app for payment

In order to simulate these scenarios, we, first of all, concentrated on the changes to be made to the configuration of the **Neighbourhood Energy Management System** so that, in addition with the management of the loads generated in the households of the neighbourhood, should take in account also the additional power requests coming from the nearby **Charge Management System**.

In order to quickly test the first extensions of the tool, we developed a simplified stand-alone version of the simulator prototype using a **Load Scheduler** Agent that at the moment simply replies to the Load requests messages coming from the **Dispatcher** Agent without any optimizations of the neighbourhood energy.

The configuration model of a typical simulation session requires both a static configuration and a dynamic configuration.

The static configuration (the **neighbourhood.xml** file – see section 4.1.1) includes the definition of the different energy actors: producers like PVs, consumers like appliances, prosumers like batteries that make up the neighbourhood system. To each of them we could associate one or more different profiles of energy production or of energy consumption. First of all, to simulate the new GreenCharge scenarios, we, surely, need to include in the configuration of the system one or more EV chargers that can be seen as sockets or charging points of a new Energy Actor: the **Charging Station**. Each socket of the **Charging Station** can behave at different times both as consumer when needs energy to charge a plugged EV plug and as a producer if the V2G technology is enabled and authorized by the user.

The dynamic configuration (the **load.xml** file – see section 4.1.2) otherwise describe the specific loads of the day that should be managed and optimized by the **Load Scheduler** of the simulator. Each load is characterized by its constraints and its flexibility. In this first scenario, for example, we need to characterize as a load each of EV charging requests (or bookings) accepted by the charging station. As described in the precedent chapter the dynamic input data that feed the simulation session (in particular the **LOAD** events) can be add using the **Graphical User Interface** of the tool, for example to simulate a specific scenario with a specific artificial distribution of the loads. The other way, that will be the most useful for evaluating the technologies and policies tested in the Pilots, will require the need of querying external sources through the use of appropriate APIs developed in the Pilots' sites.

The configuration will be described in the next section, to configure and fill a simulation session with data input.

3.2.1. Neighbourhood Configuration

Fields:

- **Neighbourhood:** represents the entire neighbourhood to simulate. The “id” is a number that identifies the neighbourhood. It contains all the houses that you want to simulate.
 - **House:** represents the household. The “id” is a number that identifies the house within the neighbourhood. It contains a single user in the current version.
 - **User:** Currently, for each house there is only one user but for possible future user we prefer to add it. The user contains all the devices of the house.
 - **Device:** represents a consumer or a producer device. It is described by:
 - **id:** it identifies the device within the house
 - **name:** a mnemonic name
 - **type:** it can be consumer or producer.
 - **HeaterCooler:** represents a Heater/Cooler unschedulable device. It is described by:
 - **id:** it identifies the device within the house
 - **name:** a mnemonic name
 - **type:** it is always N/S-Consumer
 - **BackgroundLoad:** represents a background unschedulable device:
 - **id:** it identifies the device within the house
 - **name:** a mnemonic name
 - **type:** it is always N/S-Consumer
 - **ChargingPoint:** represents an energy group type:
 - **peakload:** is the power peak of the energyGroup in W
 - **id:** it identifies the device within the house
 - **connectorsType:** Type of connector (AC/DC or both)
 - **Ecar:** represents an EV:
 - **model:** the EV model (e.g., Volkswagen e-Golf)
 - **id:** unique id for the device
 - **capacity:** Battery capacity in W.
 - **maxchpowac:** max Charging Power in AC
 - **maxchpowcc:** max Charging Power in DC
 - **maxdispowac:** max Discharging Power in AC
 - **maxdispowcc:** max Discharging Power in DC
 - **maxallen:** Max energy Allowable
 - **minallen:** Min energy Allowable
 - **sbch:** Energy threshold above which the charging efficiency decrease
 - **sbdiss:** Energy threshold above which the discharging efficiency decrease
 - **cheff:** Charging efficiency (between 0 e 1)
 - **diseff:** DisCharging efficiency (between 0 e 1)
 - **type:** Type of device
 - **name:** a mnemonic name
 - **Battery:** represents a battery
 - **model:** the battery model
 - **id:** unique id for the device
 - **capacity:** Battery capacity in W.
 - **maxchpowac:** max Charging Power in AC
 - **maxchpowcc:** max Charging Power in CC

- **maxdispowac**: max Discharging Power in AC
 - **maxdispowcc**: max Discharging Power in CC
 - **maxallen**: Max energy Allowable·minallen: Min energy Allowable
 - **sbch**: Energy threshold above which the charging efficiency decrease
 - **sbd**: Energy threshold above which the discharging efficiency decrease
 - **cheff**: Charging efficiency (between 0 e 1)
 - **dis**: DisCharging efficiency (between 0 e 1)
 - **type**: Type of device
 - **name**: a mnemonic name
- **Charging Station** represents the charging station associated to the neighbourhood. The “id” is a number that identifies the charging station within the neighbourhood.
 - **User**:
 - **Device**: represents a charging point:
 - **id**: it identifies the socket within the station
 - **name**: a mnemonic name
 - **type**: Prosumer
 - **ChargingPoint**: represents an energy group type:
 - **peakload**: is the power peak of the energyGroup in W
 - **id**: it identifies the device within the house
 - **connectorsType**: Type of connector (AC/DC or both)
 - **Ecar**: represents an EV:
 - **model**: the EV model (e.g., Volkswagen e-Golf)
 - **id**: unique id for the device
 - **capacity**: Battery capacity in W.
 - **maxchpowac**: max Charging Power in AC
 - **maxchpowcc**: max Charging Power in CC
 - **maxdispowac**: max Discharging Power in AC
 - **maxdispowcc**: max Discharging Power in CC
 - **maxallen**: Max energy Allowable
 - **minallen**: Min energy Allowable
 - **sbch**: Energy threshold above which the charging efficiency decrease
 - **sbd**: Energy threshold above which the discharging efficiency decrease
 - **cheff**: Charging efficiency (between 0 e 1)
 - **dis**: DisCharging efficiency (between 0 e 1)
 - **type**: Type of device
 - **name**: a mnemonic name
 - **Battery**: represents a battery
 - **model**: the battery model
 - **id**: unique id for the device
 - **capacity**: Battery capacity in W.
 - **maxchpowac**: max Charging Power in AC
 - **maxchpowcc**: max Charging Power in CC
 - **maxdispowac**: max Discharging Power in AC
 - **maxdispowcc**: max Discharging Power in CC
 - **maxallen**: Max energy Allowable·minallen: Min energy Allowable
 - **sbch**: Energy threshold above which the charging efficiency decrease
 - **sbd**: Energy threshold above which the discharging efficiency decrease

- **cheff**: Charging efficiency (between 0 e 1)
- **diseff**: DisCharging efficiency (between 0 e 1)
- **type**: Type of device
- **name**: a mnemonic name

3.2.2.Loads Configuration

This file (**load.xml**) should contain all the energy requests that must be dynamically satisfied and optimized by the scheduler. These requests arrive at different times during the simulation session and are characterized by a submission time (**Creation Time**), a specific profile of energy required (**Profile**) and a fixed degree of flexibility (**EST** and **LFT**). All these load requests are managed as events by the simulator and are ordered on the basis of the **Creation Time** field.

The structure of each load event should be the following:

- **Device**: it indicates the device in which the load event refers to;
- **EST**: Earliest Start Time, it's a timestamp;
- **LFT**: Latest Finish Time, it's a timestamp;
- **Creation_Time**: event creation timestamp;
- **Profile**: describes the operating model of the device and its associated to a specific energy request file (i.e., different ways of using a washing machine consume differently).

This is the structure and the parameters used by the Load Scheduler in CoSSMic to optimize the scheduling of the energy and maximize the self-consumption rate of the neighbourhood. The initial idea is to use the same structure of the event also to characterize the energy demand (**LOAD**) of an EV, modelled as a cumulative energy timeseries (**Profile**), submitted at plugin-time, with an Earliest Start Time (**EST**), that in this case is equal to the **Planned Arrival Time** and a Latest Finish Time (**LFT**) equal to the **Planned Departure Time**. In summary, for an electric car that needs to be recharged at a charging station most likely we could reasonably rely on information like this:

EV model (likely to be specific constraints on charging process associated with the different models).

This must be described in templates (or similar) per EV model

Planned Arrival Time (EST)

Planned Departure Time (LFT)

Initial State of Charge

Target State of Charge

Actual Arrival Time

Actual Departure Time

V2G enabled with any limitations or constraints, to serve as standby capacity in case of unplanned need to use the car.

The objective is therefore with a pre-processing phase to generate from this initial information one or more events with a structure similar to the event specified above, probably with an energy profile dynamically calculated through a simple analytical model of the charging process.

In this first version of the prototype, associated to a charging request from an EV we will find in the file **loads.xml** these parameters:

- Planned Arrival Time

- Planned Departure Time
- E-Car (EV) Model (type of battery)
- Initial Status of Charge
- Target Status of Charge
- V2G enabled

3.3. Graphic User Interface

The GreenCharge Simulator Graphic User Interface aims to be simple, smart and intuitive. The top menu in the first version was designed to contain 4 tabs: **Settings**, **Dynamic Settings**, **Control Panel**; **Show Results** and **GreenCharge Simulation Tool Info** (Figure 4).

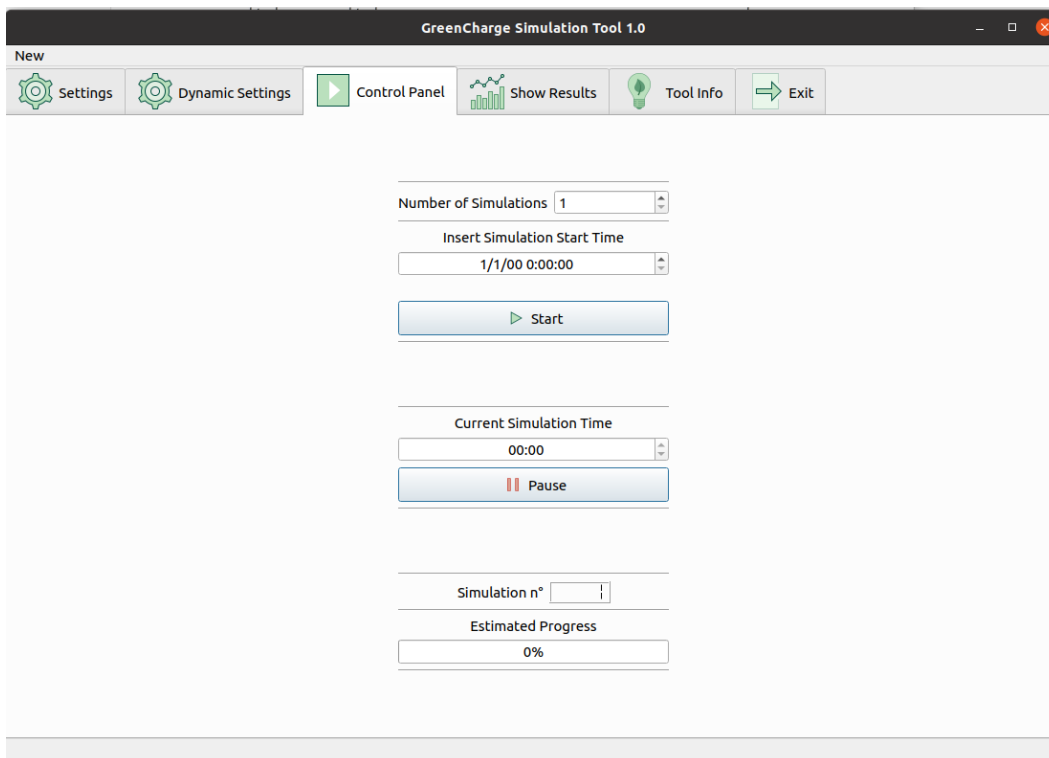


Figure 4 GreenCharge Control Panel

3.3.1.Settings

The **Settings** panel is divided into two fields. The first deals with the static and topological configuration of the neighbourhood and the second with the dynamic configuration of the simulation session where we can upload the devices together with their energy demand. In the next section we illustrate more accurately these two distinct logical steps of the configuration phase.

3.3.2.Control Panel

The **Control Panel** represents a kind of dashboard of the tool to set and overlook a simulation session (Figure 5).

In this panel, after completing the configuration phase, we can fix the day and the starting time of the simulation. Pressing the start button activates all the simulation agents and starts the scheduling process. As the simulation progresses over time, the actual simulation time is updated, allowing the user to keep track of the simulation evolution. Furthermore, the user can pause and change the current simulation time, possibly

skipping a few hours or going back in time. This feature could be very useful since the user could dynamically insert other devices or changing some parameters into the configuration in order to test, on the fly, new simulation conditions.

Finally, a progress bar indicates to the user the percentage of completion of the simulation.

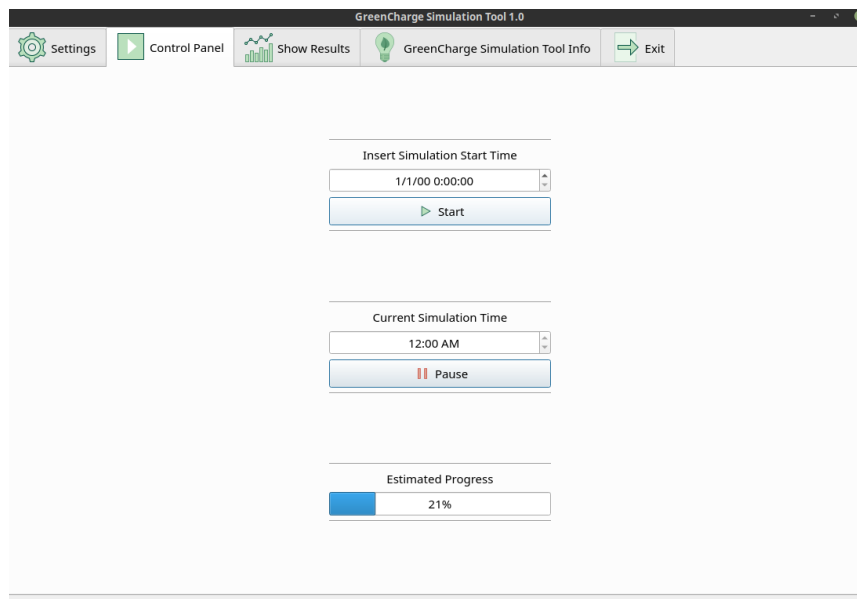


Figure 5 Simulator Dashboard

3.3.3. Show Results

In the **Show Results** panel, as expected, the user will be able to enquire the database of the output data produced by the current and past simulation sessions, select and visualizes different graphical views of results.

In this panel it should also be possible to calculate and display some of the KPIs defined to quantify the impact of some measures tested in the Pilots.

3.3.4. GreenCharge Simulation Tool Info

In this panel we will find information about the project, the partners involved and the contact references

3.4. The Configuration phase using the tool GUI

In this section, resuming the corresponding sections of the D5.2, we describe in detail the steps, using the Graphical User Interface, to configure the specific simulation session, start and control the simulation run. Besides, will be described the new features of the graphical interface.

In the Figure 6 **Error! Reference source not found.** is presented a complete Use Case Diagram showing how to configure and start a simulation session using the graphical features of the **Settings** panel of the developed GUI.

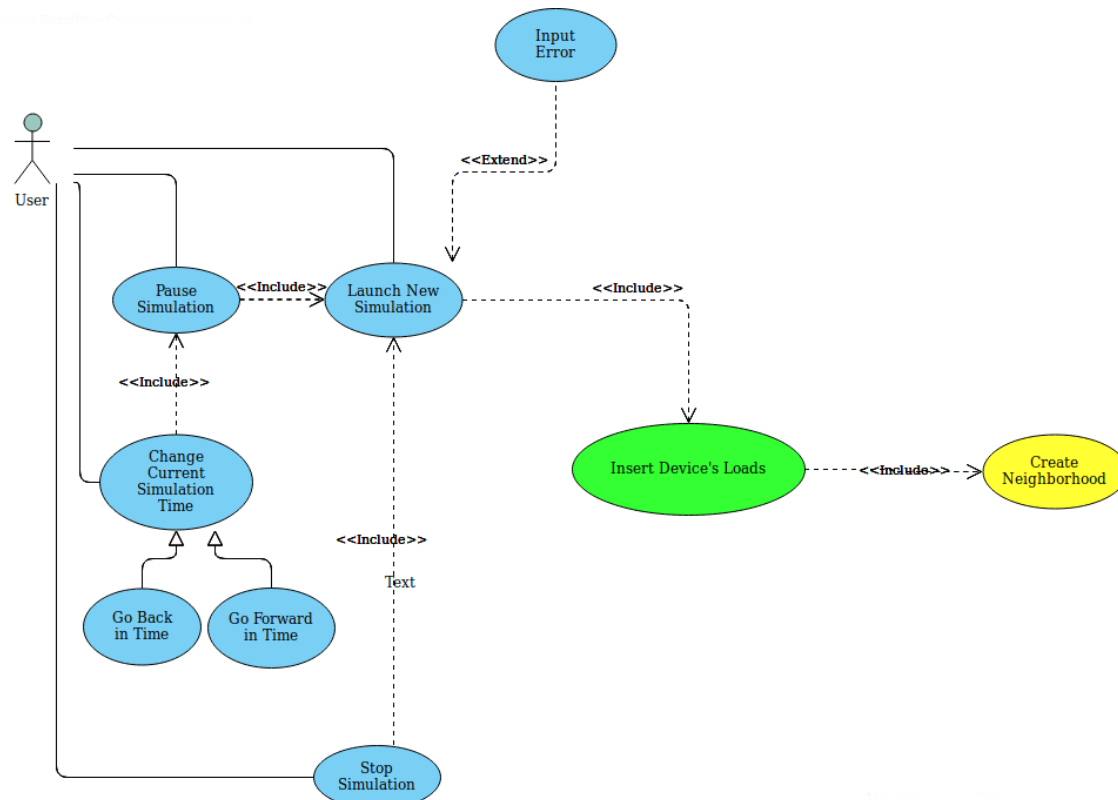


Figure 6 Setting Use Case

A user can mainly start a new simulation, pause it, stop it or change the current simulation time if it has previously paused the simulation.

However, to launch a new simulation, it is necessary to complete the configuration phase. This phase consists of two steps explained by other two use case diagrams described below:

- In the first step the user has to define the neighbourhood composition specifying the number and characteristics of the energy actors involved: charging stations, houses, together with the associated PV panels, batteries and devices (Create Neighbourhood yellow box in Figure 6).
- In the second step it is needed to set the energy requests (loads) associated to each device together with the constraints or usage preferences fixed by the user (Insert Device's Loads green box).

3.4.1. Configuration of the Neighbourhood

The first step of the simulation configuration can be done in two ways (see the Use Case Diagram Figure 7

- A user can select their own XML configuration files (neighbourhood.xml) simply by uploading an existing xml file. Once the configuration files have been selected and uploaded, the user can decide whether to modify it or leave it as it is. When a neighbourhood configuration is uploaded, also devices associated with the single neighbourhood element are loaded, so that they can be afterwards modified, added or deleted. In particular, using the GUI, it's possible to add a device already present or to create a new device from scratch.
- A user can decide to build a neighbourhood from scratch using the features of the GUI.

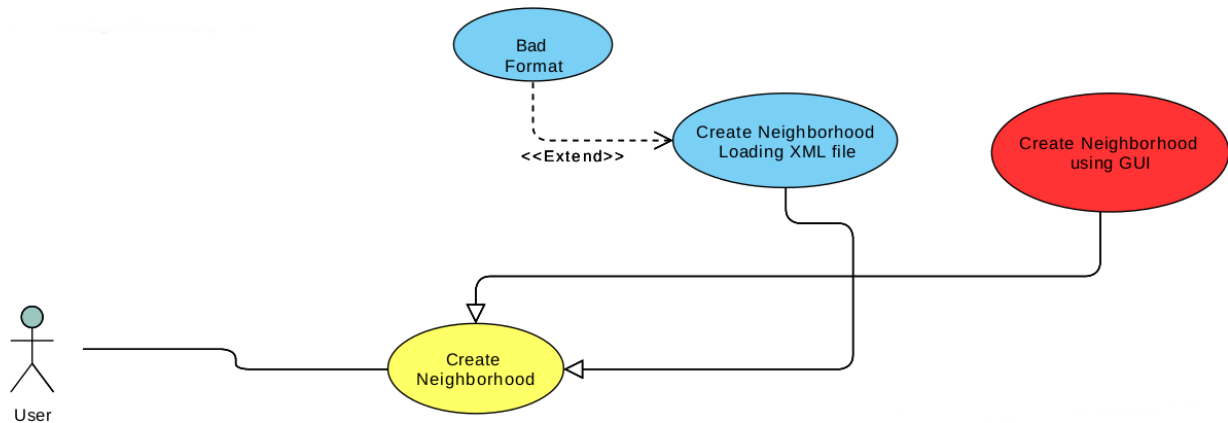


Figure 7 Create Neighbourhood Use Case a)

As said above, to associate a device or a solar panel to a house or, even, an EV to a charging station, the user can select one of the devices already defined or creates a new one (see Figure 8). Once created, the new device is added to the repository so that it can be reused in successive simulations.

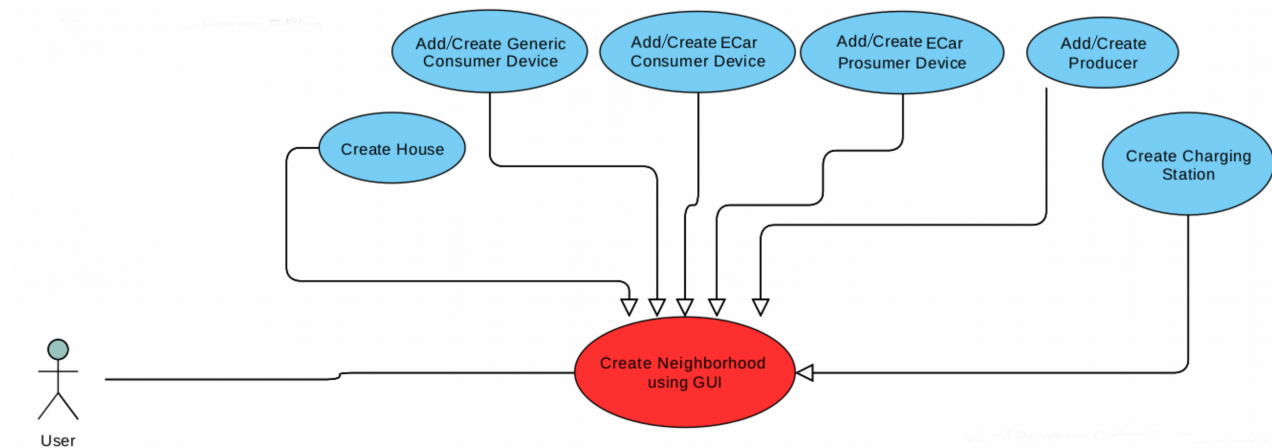


Figure 8 Create Neighbourhood Use Case b)

Furthermore, it is possible to modify a device present in the repository.

3.4.2. Definition of the Neighbourhood Loads

The **second step** can be performed in two ways (see the Use Case Diagram in Figure 9):

- A user can select their own XML configuration files (loads.xml) simply by selecting an existing xml file. As in the previous step, once the configuration file has been selected, the user can decide whether to modify the single loads or leave them as they are.
- Alternatively, the user can decide to manually add load devices to neighbourhood items. In this case it will be possible for the user to insert a load, associated to a specific device present in the configuration. *Definition of the load parameters needs to be updated according to the last version of the tool.*

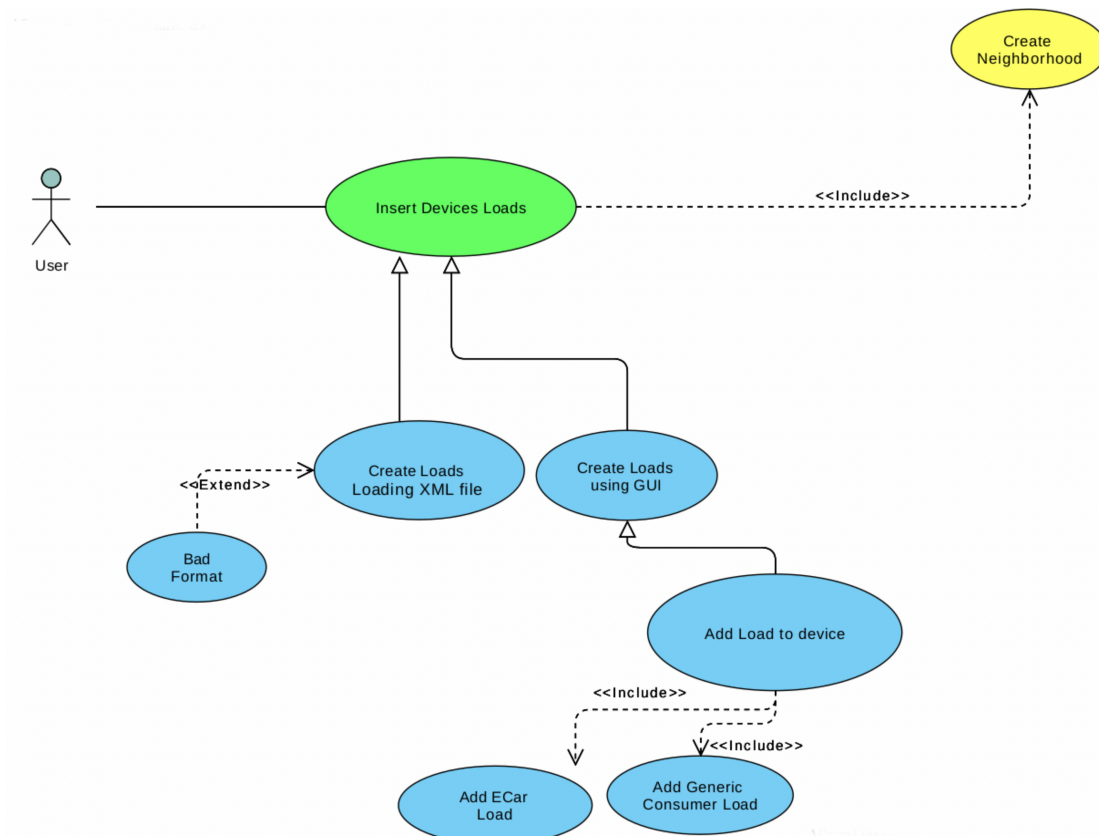


Figure 9 Insert Devices Loads Use Case

3.5. Configure and run a simulation associated with a specific scenario

The simulator user-guide is continuously updated according to the last improvements and to the new functionalities which will be integrated to better support the work of the evaluators. For this reason, a detailed documentation is provided at <https://github.com/greencharge/gcsimulator>.

In order to present to the reader a practical example of simulation with a specific scenario, we describe in Section 6 a demo, which is also provided with software for demonstration and testing purpose.

4. GreenCharge Simulator: Software Architecture and Components

4.1. Overview of the GreenCharge Simulator Software Architecture

In this section, we illustrate a sketch of the software architecture with the main components and their interactions, focusing on the following three main software components, developed specifically for the simulator prototype: **Setup Module**; **Simulation Input Data Repository**, **External Source** and **Dispatcher**.

Green Charge Simulator architecture is showed in Figure 10:

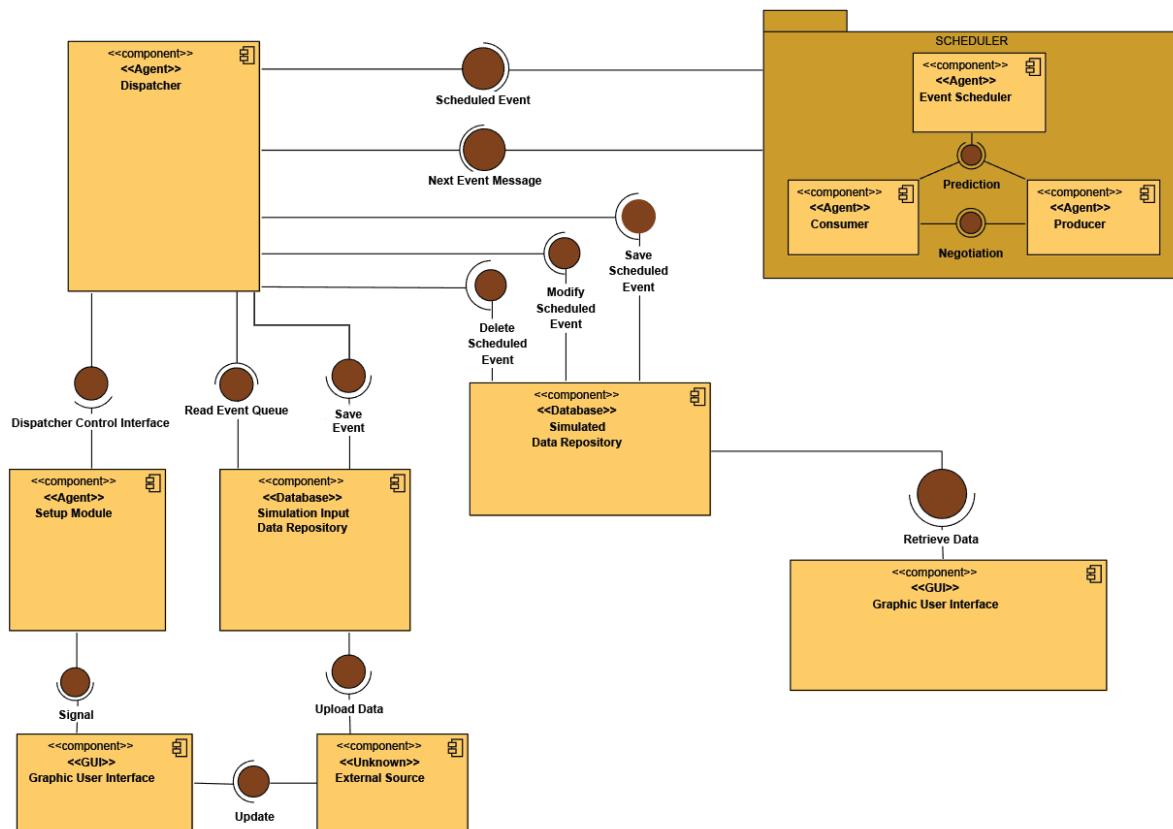


Figure 10 GreenCharge Simulator Architecture Overview

The central idea is to decouple the simulator engine, the **Dispatcher** component, that processes the events and move forward the simulated time, from the components that fills the event queue with the input data feeding the specific simulation session. Such input data, like PV panels characteristics, household's devices, energy loads etc. should be stored in a database: **Simulation Input Data Repository**. The component that has the task of filling the database is called **External Source** and can acquire the input data from different sources:

1. The GUI interface as shown in the figure above.
2. Procedures that generate a synthetic workload according to a specific statistical distribution model.
3. Directly from ad-hoc software wrappers developed in the pilots.

The ability to acquire input data in these different ways is a key aspect of the design. It offers considerable scope in setting up and experimenting with different types of simulations.

Of course, the component **External Source** that collects the data and fills the **Simulation Input Data Repository**, and the **Dispatcher** component that extracts the events and processes them, should be synchronized and supervised by another component that we call **Setup Module**. Another task assigned to this component is the possibility of stopping, pausing or restarting the simulation session.

Below we will describe the main functions of these components and their interactions, while in the next section we will give some more details on the technologies used to develop them and their implementation.

4.2. GreenCharge Simulator Components

In this section, we illustrate the role of the main software components and their functionalities together with the description of messages and protocol that governs their interactions.

The **Setup Module** is a SPADE agent that gives the start signal of a simulation session to the **Dispatcher**. Besides, the **Setup Module** catches the control events from the GUI (a user could decide to stop a simulation session pause it or restart the simulation from a new Current Simulation Time) and sends the appropriate messages to the other SPADE Agents involved in the simulation.

For example, the **Setup Module** could stop the simulation, wait for the propagation of some possible changes in the configuration to the event queue, and finally send a resume signal to the dispatcher with a new restarting time of the simulation. The simulation can start again from the point where it was stopped or from another instant, in case the user wants to go back or carry forward the current simulation time. All these signals are exchanged among the Agents using specific XMPP messages.

In the **Simulation Input Data Repository** are stored all necessary information about entities that are involved in simulation process: energy producers, consumers, and prosumers.

There's a python priority queue in which the priority is given by the event's creation time. At each get action from the queue returns the event object with the lowest value of the creation time. Different types of events are foreseen such as, for example, the request for energy from a device (**LOAD**), the update of the energy production of a panel (**UPDATE**), and so on. Most of the events processed by the simulator both for the initial configuration of the scenario and for the scheduling of energy loads will be obtained from two xml files (**neighbourhood.xml** and **load.xml**) that will be described in detail in the next chapter. There will be described the structure of some significant events that feed a simulation session, together with its specific parameters.

The **External Source** has the task of filling the **Simulation Input Data Repository** with inputs coming from different sources of data: predefined configuration files, Graphical User Interface, external data repository or third-party software API. At the end of every change in the event queue, the component notifies the **Setup Module** that the upload is terminated in order to maintain data consistency and integrity.

In this initial version of the prototype, the **External Source** is a SPADE agent with a single behaviour. Basically, at the start of a simulation session, it reads the neighbourhood structure and all devices in the neighbourhood, then it creates the event and associates them with the previously loaded devices, finally puts them in the priority queue.

In order to understand the methods of the **External Source** agent, the file and directory structure of the GreenCharge simulator is provided (Figure 11).

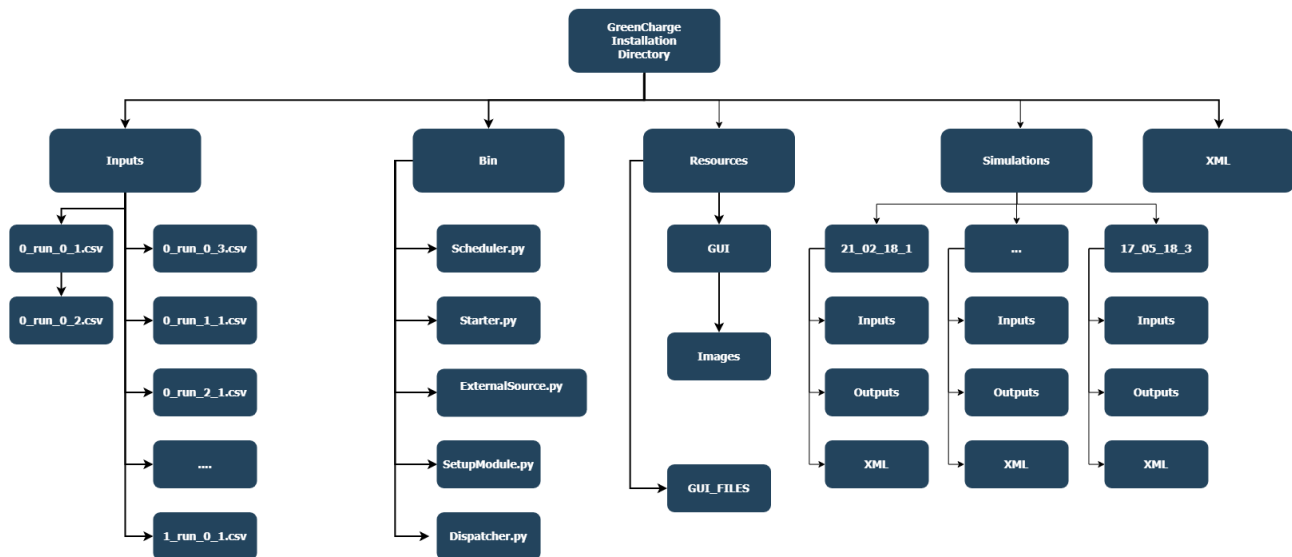


Figure 11 GreenCharge Files and Directory Tree

There are 4 main folders:

- **Inputs:** all the profiles in csv format are saved in the **Inputs** folder. All files are saved according to the format: Houseid_DeviceID_ProfileID.csv;
- **Bin:** the **Bin** folder contains all the python files needed for the simulation, including agents.
- **Resources:** all the files connected to the graphic interface are saved in the **Resources** folder, such as XML files and images present in the GUI.
- **XML:** in the **XML** folder the temporary files **neighbourhood.xml** and **loads.xml** that are built using the **GUI** are saved. When the simulation starts, they are copied to the XML subfolder of the related simulation specific folder.
- **Simulations:** in the **Simulations** folder a directory is saved for each simulation performed. The name for each folder is: **Day_Month_Year_SimulationNumberInThatDay**. In each folder are created three subfolders: in **Inputs** the csv profiles related to that simulation are saved; in **XML** are copied the xml files related to that simulation; in the subfolder **Outputs** will find the simulation results.

At the moment **External Source** executes sequentially the following four methods:

- ***makeNewSimulation ()*:**
this method creates the necessary folders for the single simulation, copies the XML files from the temporary folder to the simulation subfolder and copies the profiles necessary for the simulation to the inputs subfolder.
- ***createDevicesList ()*:**
this method reads the **neighbourhood.xml** file and creates the list of devices that will be part of the simulation.
- ***createEventList ()*:**
This method reads the **loads.xml** file and creates a list of events.
- ***uploadInInputRepository ()*:**
This method updates the priority queue by entering the events not yet processed. It is called at the beginning of the simulation and every time the user makes some changes to the runtime configuration.

The **Dispatcher** is the real Simulation engine since it controls and manages the time's flow of simulation.

Its usual operation cycle can be summarized in the following steps:

- reads next event from **Simulation Input Data Repository** relying on a current simulation time (CST);
- deliver a specific message to the **Load Scheduler** (on the basis of the next event to process);
- wait for a reply message;
- store scheduled events in **Simulated Data Repository** that contains the complete trace of the simulated session.

It's SPADE agent with two behaviours. First behaviour waits and handles **Setup Module** XMPP messages about start and stop signal. Second behaviour becomes active when a start signal is notified and it gets and consumes the next event in the priority queue. It generates different XMPP messages (depending on the specific event type) to send to the **Load Scheduler**.

In addition, the Agent **Dispatcher**, on the basis of the scheduling answer for the specific load calculated by the **Load Scheduler**, creates and inserts in the priority queue a **DELETE** load that represents the completion of the specific energy consumption period.

The detailed description of the protocol between the Simulation Engine component (the **Dispatcher**) and the **Load Scheduler**, together with the specification of the exchanged messages payload will be illustrated in the next sections.

The software architecture, above described, assumes an implementation of both the simulation engine (**Dispatcher**) and the optimiser (**Load Scheduler**) based on the agent model. This made it possible to carry out the exchange of messages and the associated protocol using the XMPP standard.

However, at a certain point in the project, in order to be able to carry out a more complete and more meaningful evaluation of the simulation results, it seemed appropriate to try to integrate in the simulation tool different optimisers that applied different scheduling policies. This required a further implementation effort to allow the simulation engine to interact both with the Agent based optimiser developed by the University of Oslo through XMPP messages, and with the optimiser developed by the partner Eurecat, using the HTTP based REST interaction model.

In the next section we will describe in more detail how the two different interaction paradigms were used in the simulator and will be presented a Web tool specifically developed to facilitate and test the integration of the two different optimization tools.

4.3. EMS Interaction Protocol

This EMS interaction protocol allows the integration of the GreenCharge Simulator with a GreenCharge Energy Management Systems (EMS). It defines the message format exchanged between the Simulator and the EMS.

Simulation Scenario

The tree shown in the main cell includes one household and the list of devices.

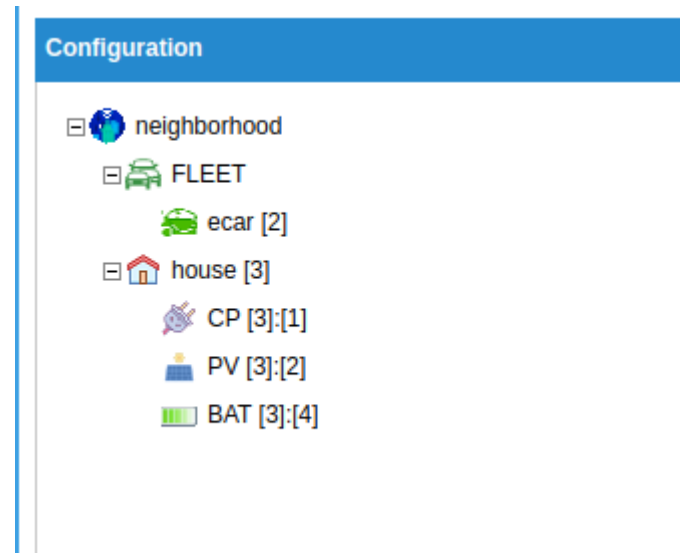


Figure 12 Simulation Scenario: configuration of the neighbourhood.

In Figure 12 smart neighbourhood is represented as a tree whose root is the grid and the devices are the leaves. Leaves of the tree can be energy consumers, producers or energy storages. Each branch includes the entities across which the energy flows (Energy Groups). At the abstract level Energy Groups can be households, charge stations and any electric panel or transformer along the line.

Each node of the tree is identified by a string, which is composed by a variable number of integers between square brackets and separated by colon.

E.g., [1]:[3]:[1] represents a device after two Energy Groups ([1] and [1]:[3]).

Exchanged Messages

The protocol is based on the exchange of five kinds of messages:

- Configuration Messages (from the Simulator to the EMS): they are sent at the beginning of the simulation. They do not cause any advance of the simulation time.
- Event Notifications (from the Simulator to the EMS): they are sent by the simulator to the EMS. They notify the occurrence of an event to the EMS. They eventually advance the current simulation time. They can be blocking messages, that means the Simulator waits for an EMS response to continue the simulation, or not.
- Synchronization Messages (from the EMS to the Simulator): they are eventually used to notify the complete the reaction to an event and allows to move forward the simulation time.
- Event Responses (from the EMS to the Simulator): they notify the reaction to an event by the EMS, such as a schedule update or any energy related information. They eventually move forward the simulation time.

Error Handling (either from the EMS or the Simulator): they are used to notify an error condition.

Each message starts with a label that defines the message type and terminates with the current simulation time, expressed as Linux epoch.

Configuration Messages

ENERGY_GROUP

This message defines a transformer, which limits the power can be drained by grouped entities. Entities after an energy groups can be households, Charge Points or each kind of devices. Then, an energy group can be configured at any point of the grid network. No answer is expected.

CREATE_ENERGY_GROUP ID POWERPEAK SIMULATION_TIME

CREATE_ENERGY_GROUP is the message type

id represent the device (e.g., a household level or a neighbourhood level transformer)

powerpeak is the power limit the energy group can consume in Watt

EXAMPLE:

```
CREATE_ENERGY_GROUP [1] 3000
```

The first message from the simulator will be always of this kind with ID [0] and represents the Grid of the neighbourhood.

CREATE_PRODUCER

This message defines the existence of a PV panel and provide a prediction of production.

It is not blocking, it is not expected a specific response. It requires at least one following message PREDICTION_UPDATE that provides a timeseries with predicted values of power production

CREATE_PRODUCER_PV ID SIMULATION_TIME

1. CREATE_PRODUCER_PV defines the message type.
2. ID identifies the Photovoltaic Panel 2 of household 0.
3. The simulation time is the last parameter.

Example:

```
CREATE_PRODUCER_PV [0]:[2] 1449885600
```

ENERGY_COST

This message defines the energy cost for the neighbourhood if the ID is [0], of any other producer with different IDs. It is sent at the beginning of simulation. It does not wait for a response, so it is not blocking for the simulator.

The message format follows:

ENERGY_COST ID PROFILE SIMULATION_TIME

ID correspond to the producer, i.e., a PV panel. It is [0] to identify the grid.

PROFILE is a link

The cost is defined as a time-series in a csv file.

The value corresponds to Eur per kWh.

Example:

```
ENERGY_COST [1]:[1] http://parsec2.unicampania.it/~gcdemo/Simulations/eurecat/energy_cost.csv
SIMULATION_TIME
```

ENERGY MIX

This message defines the energy mix for the neighbourhood. It is sent at the beginning of simulation. It does not wait for a response, then it is not blocking for the simulator.

The message format follows:

ENERGY_MIX ID PROFILE SIMULATION_TIME

ID is the identifier of the producer.

PROFILE is a link

The energy composition is defined as a time-series in a csv file that describes the percentage of black/grey/green energy delivered at a certain time. The CSV file will have four columns: The time stamp (integer), the black part (real number), the grey part (real number), the green part (real number). The sum of the real numbers should be unity (1.0)

Example:

ENERGY_MIX http://parsec2.unicampania.it/~gcdemo/Simulations/eurecat/energy_cost.csv

CREATE_EV

This message defines the static parameters of an EV belonging to the simulation scenario. It is sent at beginning of the simulation. An example of message is shown below.

**CREATE_EV ID CAPACITY MAX_CH_POW_AC MAX_CH_POW_CC MAX_DIS_POW_CC
MAX_DIS_POW_AC MAX_ALL_EN MIN_ALL_EN SB_CH SB_DIS CH_EFF DIS_EFF V2G
SIMULATION_TIME**

The message includes static parameters for the EV

1. **CREATE_EV** is the message TAG.
2. 665 is an ID, it is a unique identifier of the EV
3. 24 capacity (kWh): it is the maximum amount of energy can be stored.
4. 4 is the max_ch_pow_ac (kW): max charging power when alternate current is used
5. 40 is the max_ch_pow_cc (kW): max charging power when continuous current is used
6. 4 mac_dis_pow_cc (kW): max discharging power
7. 40 mac_dis_pow_ac (kW): max discharging power
8. 99 is the max_all_en: (%) maximum allowed energy can be stored as a percentage of the capacity
9. 2 is the min_all_en: (%) minimum allowed energy must be stored as a percentage of the capacity
10. 90.0 sb_ch: (%) energy threshold above which the charging efficiency decrease
11. 10.0 sb_dis: (%) energy threshold below the discharging efficiency decrease
12. 0.7 ch_eff: ([0,1]) this parameter must be multiplied to the nominal maximum charging power to obtain the actual maximum charging power above sb_ch.
13. 0.7 dis_eff: ([0,1]) this parameter must be multiplied to the nominal maximum discharging power to obtain the actual maximum discharging power below sb_dis.
14. 0/1 V2G defines if the EV support V2G energy transfer

Example:

CREATE_EV [665] 24 4 40 4 40 99 2 90.0 10.0 0.7 0 1449885600

CREATE_BATTERY

This message defines the creation of a battery.

```
CREATE_BATTERY ID CAPACITY MAX_CH_POW_AC MAX_CH_POW_CC MAX_DIS_AC
MAX_DIS_CC MAX_ALL_EN MIN_ALL_EN SB_CH SB_DIS CH_EFF DIS_EFF
SOC_AT_ARRIVAL START_TIME END_TIME TARGET_SOC SIMULATION_TIME
```

It contains both static and dynamic parameter, as for now it is supposed that the battery exists from the beginning to the end of the simulation.

Static Parameters include:

1. ID: a unique identifier of the battery (es: [0]:[1])
2. capacity (kWh): it is the maximum amount of energy can be stored.
3. max_ch_pow_ac (kW): max charging power when alternate current is used
4. max_ch_pow_cc (kW): max charging power when continuous current is used
5. max_dis_pow_ac (kW): max discharging power
6. max_dis_pow_cc (kW): max discharging power
7. max_all_en: (%) maximum allowed energy can be stored as a percentage of the capacity
8. min_all_en: (%) minimum allowed energy must be stored as a percentage of the capacity
9. sb_ch: (%) energy threshold above which the charging efficiency decrease
10. sb_dis: (%) energy threshold below the discharging efficiency decrease
11. ch_eff: ([0,1]) this parameter must be multiplied to the nominal maximum charging power to obtain the actual maximum charging power above sb_ch.
12. dis_eff: ([0,1]) this parameter must be multiplied to the nominal maximum discharging power to obtain the actual maximum discharging power below sb_dis.

Dynamic Parameters include:

1. Soc_at_arrival: (%) the amount of energy stored in the battery as a percentage of capacity
2. Start_time: (datetime) the time in wich the battery is inserted into the system
3. End_time: (datetime) the time in wich the battery is removed by the system
4. Target_soc: (%) the amount of energy that it desired to be stored at the end time

EXAMPLE:

```
CREATE_BATTERY [0]:[6] 24 4 40 99 2 90.0 0.7 5 1449925200 1449914400 72
```

SYNCHRONIZATION MESSAGES

SCHEDULED

This is an event response that alerts dispatcher to continue sending messages. It is sent when all reactions to a blocking event notification have been terminated.

```
SCHEDULED EXECUTION_TIME
```

Time represents CPU time of the optimization as decimal seconds

EXAMPLE:

```
SCHEDULED 1449922200
```

SIMULATION END

This message communicate that the simulation is terminated, that means there are no requests in the simulator queue. When the simulator receives this message all response must be returned and the simulator must be stopped by the scheduler itself.

SIMULATION_END 1449932400

N.B. : After this message the Simulator still runs, waiting for additional messages that eventually will be received from the EMS. It needs to be terminated explicitly by stopping the application.

EVENT NOTIFICATIONS

BACKGROUND LOAD

This message defines an energy profile consumption of the household that cannot be scheduled.

No answer is expected. It includes 3 parameters plus the simulation time

BACKGROUND_LOAD ID Sw EQ PROFILE SIMULATION_TIME

1. **BACKGROUND_LOAD** is the message type
2. **ID** represents the device (e.g., main meter 5 of househod 0)
3. **SEQ** is an incremental number and defines the i-th message from the the same device
4. **PROFILE** is n URL of the timeseries that defines cumulative consumed energy
5. The simulation time completes the message.

Example:

BACKGROUND_LOAD [0]:[5] 0
http://parsec2.unicampania.it/~gcdemo/Simulations/eurecat/12_12_15_114/0_run_0_1_ein.csv
1449874800

The profile is a time-series, containing cumulative energy consumption (kWh).

1475962935 0.007824
1475963050 0.008074
1475963201 0.009548

HEATING_COOLING_LOAD

This message defines the profile of a Heating Cooling device.

It requires an HC_PROFILE response. It is a not blocking message, it means that the response can be sent at the end of the simulation. It includes 5 parameters.

HEATING_COOLING_LOAD ID SEQ NOMINAL_POWER SIMULATION_TIME

An example follows:

HEATING_COOLING_LOAD [0]:[4] 0 [500, 1000, 1500]
http://parsec2.unicampania.it/~gcdemo/Simulations/eurecat/12_12_15_114/hcprofile.csv 1449874800

1. **HEATING_COOLING_LOAD** defines the message type.
2. **ID [0]:[4]** identifies the device 4 of the household 0.
3. **SEQ 0** is an incremental number that change when further messages are sent for the same device.
4. **NOMINAL_POWER [500, 1000, 1500]**: it is an array of values that represent the power consumed by the device in the corresponding different working mode.
5. The **URL** points to a time-series that defines the flexibility in terms of cumulative energy consumption of the device

The profile is a time series containing an average cumulative energy and a deviation that can be accepted for the consumption profile of the device is accepted.

```
1475877600 349.511604 0.006453
1475878538 349.521398 0.003260
1475879477 349.529164 0.004613
1475880416 349.532433 0.004648
1475881354 349.538647 0.004076
1475882293 349.547011 0.004365
1475883232 349.552792 0.004508
1475884170 349.557299 0.004681
1475885109 349.560741 0.004572
1475886048 349.569669 0.002791
1475886986 349.578870 0.004478
```

The response must contain a time-series with the actual consumption.

PREDICTION UPDATE

This message defines the prediction update of a PV panel and provide an updated prediction of production. This message could trigger a rescheduling, and updates can be sent to the simulator, than it blocks the simulator until a SCHEDULE message is received.

PREDICTION UPDATE ID PROFILE SIM_TIME

An example follows:

PREDICTION UPDATE [0]:[2]
http://parsec2.unicampania.it/~gcdemo//Simulations/eurecat/12_12_15_114/0_2_prediction.csv
1449885600

1. **PREDICTION_UPDATE** defines the message type.
2. **ID [0]:[2]** identifies the Photovoltaic Panel 2 of household 0.
3. An **URL** refer to the timeseries of predicted energy production.

The profile is a timeseries containing cumulative energy production (kWh).

```
1449906065 0.000000
```

1449906245 5.000000
1449906425 10.000000
1449906605 10.000000
1449906785 15.000000
1449906965 20.000000
1449907145 25.000000
1449907325 30.000000
1449907505 35.000000
1449907685 40.000000
1449907865 45.000000
1449908045 55.000000
1449908225 60.000000
1449908405 65.000000
1449908585 75.000000

EV

This message defines any events related to an electric vehicle. This message is sent two or more times and according to the parameter values can refer to a charge booking request, a booking update, the arrival of an EV or its departure.

The message format is:

EV ID SOC_AT_ARRIVAL DEPARTURE__TIME ARRIVAL_TIME CP V2G TARGET_SOC PRIORITY SIM_TIME
--

An example is shown below.

EV [665] 5 1449925200 1449914400 [1]:[0] 0 72 0 1449907200

The parameters of the message are:

1. **ID [665]** is the unique identifier of the EV into the neighborhood
2. **5 Soc_at_arrival:** (%) the amount of energy stored in the EV battery as a percentage of capacity
3. **1449925200 departure_time:** (datetime) the planned departure time communicate at the time of booking
4. **1449914400 arrival_time:** (datetime) the actual arrival time that is known when the car is plugged in at the charge point
5. **[1]:[0] charging_point:** defines the charging point where the EV will be/is connected. If the ID contains the value -1 (E.g., [-1]:[-1]) it means that is up to the scheduler to decide to which CP the EV will be connected.
6. **0 V2G:** (boolean) It states if the car support V2G capability and the driver allows the charge point to exploit it
7. **72 Target_soc:** (%) the amount of energy that must be stored at the planned departure time

8. 0 is the **priority level** for charging
9. 1449907200 **Simulation time**

The first time this request is sent at booking time (if the charge has been booked). Multiple times before the arrival time, a new instance of this request corresponds to a booking update.

If the arrival time greater than the simulation time, than the schedule receives a booking update.

If the simulation time is equal to the arrival time, the EV message triggers the arrival event.

If the simulation time is greater than the arrival time that the EV message triggers the departure event for that EV.

In the theoretical case, when the EV arrives and leaves in time with the required charge the three message are the same, only the simulation time changes.

Summarizing:

- $SIM_TIME < ARRIVAL_TIME \rightarrow$ it is a booking request
- $SIM_TIME = ARRIVAL_TIME \rightarrow$ the EV has arrived
- $SIM_TIME = DEPARTURE_TIME \rightarrow$ the EV is departed

LOAD

This message defines any shiftable load (i.e., a washing machine or a dishwasher)..

An ASSIGNED_START_TIME response is expected.

It is blocking for the simulator. It waits for at least a response. However, a Load can trigger a rescheduling for some loads. For this reason, it waits until a SCHEDULED message has been received. According to the defined protocol.

```
LOAD [0]:[1]:[1] 1 1449921600 1449932400
http://parsec2.unicampania.it/~gcdemo//Simulations/eurecat/12_12_15_114/0_run_4_1_ein.csv
1449910800
```

1. **LOAD**: defines the message type
2. [0]:[1]:[1] identifies the profile 1 of device 1 of household 0.
3. **SEQUENCE** 1 is an incremental number that is used to distinguish several execution of the same device
4. **EST** 1449921600 is the earliest start time for the device
5. **LST** 1449932400 is the latest start time for the device.
6. **PROFILE** is the cumulative energy timeseries for this device

The profile is a timeseries with relative time and cumulative consumed energy

```
0 0.000000
38 0.000250000
39 0.000256579
40 0.000256579
```

41 0.000256579
46 0.000256579
47 0.000256579
48 0.000256579
49 0.000256579
151 0.000666579
299 0.001811481
617 0.004083238
[...]

DELETE LOAD

This message is sent when a device execution is terminated. It needs to report the amount of energy which has been actually consumed.

The message format is shown below:

```
DELETE_LOAD ID CONSUMED_ENERGY SIM_TIME
```

An example follows:

```
DELETE_LOAD [0]:[1]:[1] 0.569 449932400
```

DELETE_LOAD defines the message type.

[0]:[1]:[1] identifies the terminated device

0.569 is the amount of consumed energy (kWh) 449932400 is the simulation time

EVENT RESPONSES

ASSIGNED START TIME

This message is sent from the EMS to the Simulator as a response to a LOAD message, but can be sent again also if any other event require a re-scheduling of referred load.

This message implies that the simulator will send a DELETE request when the load terminates.

```
ASSIGNED_START_TIME [0]:[1]:[1] 1 1449922200
```

1. ASSIGNED_START_TIME defines the message time-based
2. [0]:[1]:[1] identifies the device to starting
3. 1 is the execution sequence number
4. 1449922200 is the start time assigned by the scheduler,

HC PROFILE

This is an event response, by which the EMS uploads the HC profile time-series.

It is a multipart message that includes a form field (with the message label and the message parameters) and contains a CSV file attached.

The template of the form field is

HEATING_COOLING_PROFILE ID FILENAME SIMULATION_TIME

where:

1. ID: is the identifier of the HC device
2. FILENAME: is the file name of the attached CSV filename
3. SIMULATION_TIME: is the current simulation time at which the profile has been computed

It is implemented as an HTTP Post request. Here it follows an example of raw HTTP communication.

Request Header

POST /postanswer HTTP/1.1

Host: parsec2.unicampania.it:10024

Connection: keep-alive

Content-Length: 1299

Pragma: no-cache

Cache-Control: no-cache

Accept: */*

User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36

Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryaZZNRX9qdz0DLAM9

Origin: http://parsec2.unicampania.it

Referer: http://parsec2.unicampania.it/~gcdemo/demo/xmppscheduler/index.php

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.9,it;q=0.8,sm;q=0.7

Form Data

-----WebKitFormBoundaryaZZNRX9qdz0DLAM9

Content-Disposition: form-data; name="thefile"; filename="test.csv"

Content-Type: application/octet-stream

1475877600 349.511604

1475878538 349.521398

1475879477 349.529164

1475880416 349.532433
1475881354 349.538647
1475882293 349.547011
1475883232 349.552792
1475884170 349.557299
[...]

-----WebKitFormBoundaryZZNRX9qdz0DLAM9
Content-Disposition: form-data; name="response"

{"subject":"HC_PROFILE","time":"1449874800","id":["1":[2]}
-----WebKitFormBoundaryZZNRX9qdz0DLAM9--

EV PROFILE

This is an event response, by which the EMS uploads the EV profile time-series.

It is a multipart messages that includes a form field (with the message label and the message parameters) and contains a CSV file attached.

The template of the form field is

EV_PROFILE ID FILENAME SIMULATION_TIME

where:

1. ID: is the identifier of the EV device
2. FILENAME: is the file name of the attached CSV filename
3. SIMULATION_TIME: is the current simulation time at which the profile has been computed

It includes the charged profile of the EV as an attachment of the multipart http body.

An example of profile is:

1449914400,1.2 1449925200,13.2
1449914400,1.2 1449925200,13.2

POST /postanswer HTTP/1.1

Host: parsec2.unicampania.it:10024

Connection: keep-alive

Content-Length: 358

Pragma: no-cache

Cache-Control: no-cache

Accept: */*

User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36

Content-Type: multipart/form-data; boundary=----WebKitFormBoundarym9iKU4ejuvBfPHwK

Origin: http://parsec2.unicampania.it

Referer: http://parsec2.unicampania.it/~gcdemo/demo/xmppscheduler/index.php

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.9,it;q=0.8,sm;q=0.7

-----WebKitFormBoundarym9iKU4ejuvBfPHwK

Content-Disposition: form-data; name="response"

{"subject":"EV_PROFILE","id":"5"}

-----WebKitFormBoundarym9iKU4ejuvBfPHwK

Content-Disposition: form-data; name="thefile"; filename="test.csv"

Content-Type: application/octet-stream

1449921600 150

1449932400 158.4

-----WebKitFormBoundarym9iKU4ejuvBfPHwK--

Refer the same Response in the http protocol.

BATTERY PROFILE

This is an event response, by which the EMS uploads the EV profile time-series.

It is a multipart messages that includes a form field (with the message label and the message parameters) and contains a CSV file attached.

The template of the form field is:

BATTERY_PROFILE ID	FILENAME	SIMULATION_TIME
--------------------	----------	-----------------

Protocol Implementation

Two versions of the interaction protocol have been currently implemented.

- The first version uses as transport mechanism the XMPP protocol for the message exchange and the HTTP only to download and upload energy profiles. The XMPP body contains the message string as it has been defined in the previous section. The order of parameters is fixed.
- The second version uses only the HTTP protocol. Each message contains a json form and eventually a file attached in the case of upload of energy profile.

Implementation details, and future updates, are provided by the Developer Guide of the Simulator available at <http://github.com/greencharge/gcsimulator>

Integration Tool

In order to support the developer for implementing and testing the interaction of a new EMS with the Simulator we developed a web tool.

The tools work as a web interface of the GreenCharge simulator, but with limited functionalities respect the standalone Simulator GUI presented in Section 3.2.

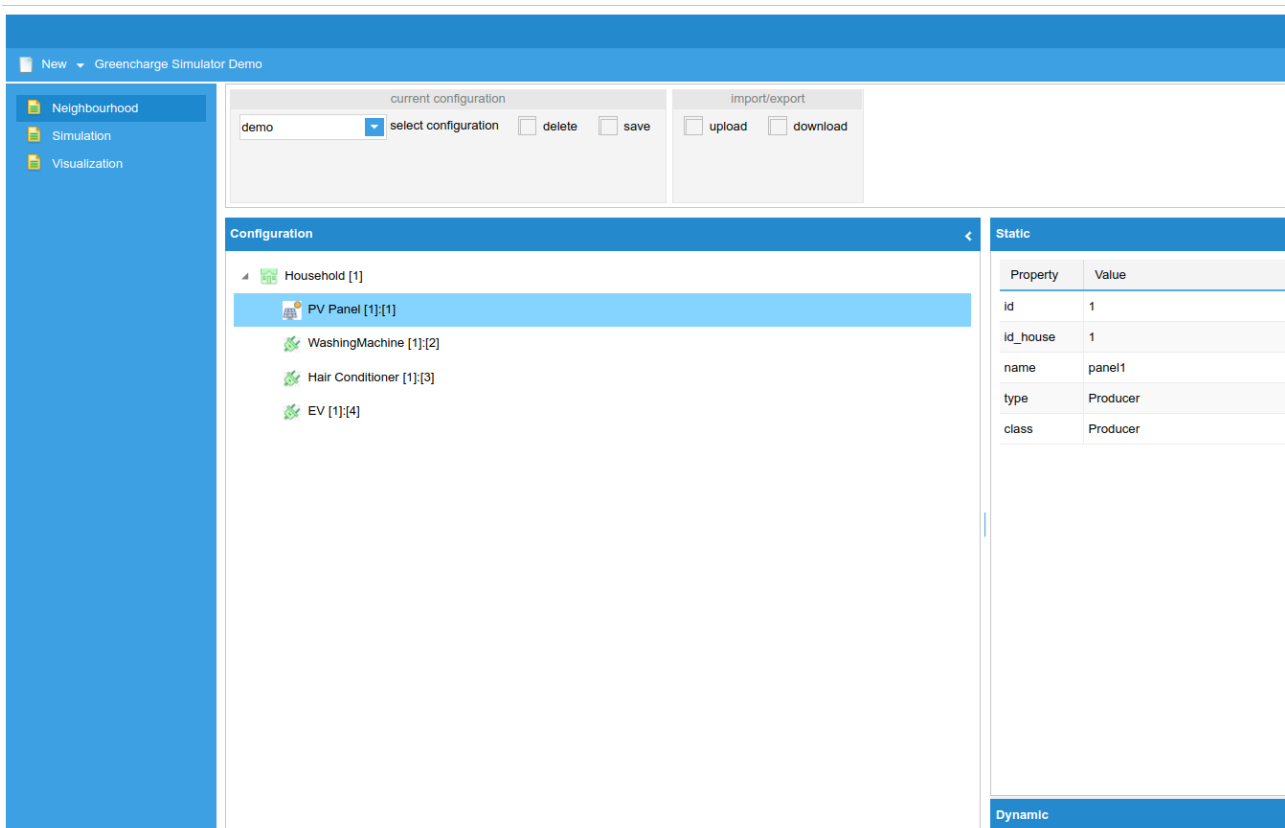
Authentication

Currently the authentication is allowed by an email and a password.

Registration and external mechanisms are disabled. To use the tool an account must be provided by SUN to the developers.

Simulation Scenario

The first dashboard is shown in Figure 13 visualizes the configuration of the simulation. It represents the configuration of an Energy Smart Neighbourhood. At the state of the art, for the integration test, only one configuration is allowed. It includes one household and four devices. The tree shown in the main cell includes one household and the list of devices. The tables on the right cells include static and dynamic properties of devices.



The screenshot shows the 'Greencharge Simulator Demo' interface. On the left is a sidebar with 'Neighbourhood', 'Simulation', and 'Visualization' tabs. The main area is titled 'Configuration' and shows a tree structure for 'demo'. The tree includes a 'Household [1]' which contains a 'PV Panel [1]:[1]', 'WashingMachine [1]:[2]', 'Hair Conditioner [1]:[3]', and 'EV [1]:[4]'. The 'PV Panel [1]:[1]' is selected. On the right, a 'Static' table displays properties for the selected device:

Property	Value
id	1
id_house	1
name	panel1
type	Producer
class	Producer

Below the static table is a 'Dynamic' section, which is currently empty.

Figure 13 Main dashboard of the integration tool.

Simulation execution

The second dashboard shown in Figure 14 allows to configure, start and stop the simulator. In particular the user can select the communication protocol: XMPP or HTTP (REST). The text-fields show the parameter required to connect the EMS to the simulator.

The “Web EMS” blue button opens a web EMS that has been implemented to provide a working example of the interaction protocol. The Web EMS does not run an optimization protocol but returns trivial responses.

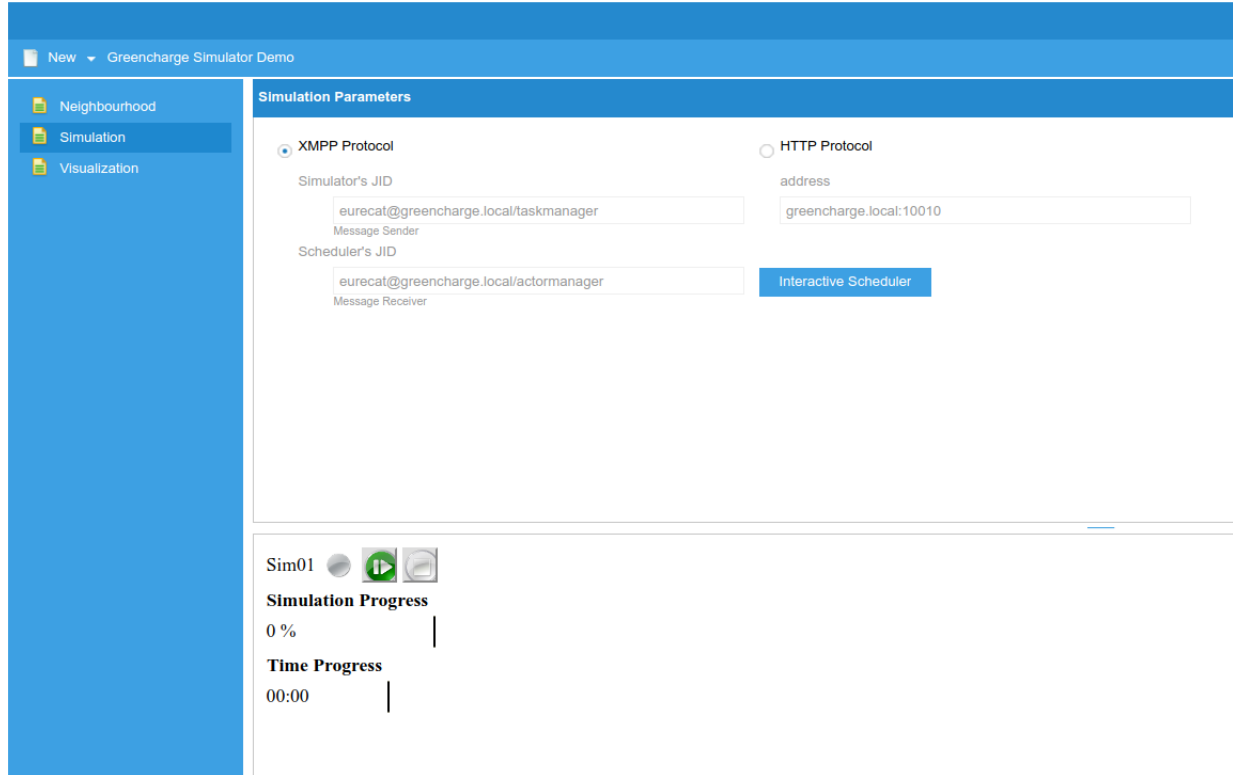


Figure 14 Simulation dashboard.

For the XMPP protocol:

- The XMPP server is waiting on port 5222 at parsec2.unicampania.it
- The Simulator's JID is needed to send message to the simulator
- The Scheduler's (EMS) JID must be used by the EMS to connect and receive message from the simulator. The password will be the same used for the authentication at login of the web tool.
- The HTTP address is relevant to upload files via multipart http POST requests.

For the HTTP protocol:

- only the http address is relevant, as each instance of the simulator use a different port and expose a rest interface that allow to get or post messages.

The cell at the bottom of the page allow to start and stop the simulator. The circle becomes red or green according to the status of the simulator.

Web EMS

In order to provide a reference implementation and to allow to the developer to look at the raw messages exchanged between the Simulator and the EMS the tools integrates two EMSs, which are embedded in a web page and have been implemented in JavaScript.

The web page of the XMPP EMSs, is shown in Figure 15.

Interactive Scheduler

JID:

Password:

☒ auto

Strophe is connecting.

Strophe is connected.

Message from eurecat@parsec2.unicampania.it/taskmanager: BG ID [0]:[0] SEQ 0
http://parsec2.unicampania.it/~gcdemo/Simulations/12_12_15_78/0_run_6_1.csv 1449874800

Message from eurecat@parsec2.unicampania.it/taskmanager: HC ID [0]:[4] SEQ 0
http://parsec2.unicampania.it/~gcdemo/Simulations/12_12_15_78/1_run_5_1.csv 1449874800

Message from eurecat@parsec2.unicampania.it/taskmanager: CREATE_PRODUCER PV [0]:[2]
http://parsec2.unicampania.it/~gcdemo/Simulations/eurecat/12_12_15_78/6_run_3_1_eout.csv 1449885600

Message from eurecat@parsec2.unicampania.it/taskmanager: EV_BOOKING CAPACITY 24 MAX_CH_POW_AC 4 MAX_CH_POW_CC 40 MAX_ALL_EN 99
MIN_ALL_EN 2 SB_CH 90.0 CH_EFF 0.7 SOC_AT_ARRIVAL 5 PLANNED_DEPARTURE_TIME 1449925200 ARRIVAL_TIME 1449914400 V2G 0 TARGET_SOC 72

Message from eurecat@parsec2.unicampania.it/taskmanager: LOAD ID [0]:[1]:[1] SEQUENCE 1 EST 1449921600 LST 1449932400 PROFILE
http://parsec2.unicampania.it/~gcdemo/Simulations/eurecat/12_12_15_78/0_run_3_1.csv 1449910800

sent response:ASSIGNED_START_TIME [0]:[1]:[1] 1 1449922200 PV [0]:[0]

Message from eurecat@parsec2.unicampania.it/taskmanager: EV_ARRIVAL CAPACITY 24 MAX_CH_POW_AC 4 MAX_CH_POW_CC 40 MAX_ALL_EN 99
MIN_ALL_EN 2 SB_CH 90.0 CH_EFF 0.7 SOC_AT_ARRIVAL 5 PLANNED_DEPARTURE_TIME 1449925200 ARRIVAL_TIME 1449914400 V2G 0 TARGET_SOC 72

Message from eurecat@parsec2.unicampania.it/taskmanager: EV_DEPARTURE CAPACITY 24 MAX_CH_POW_AC 4 MAX_CH_POW_CC 40 MAX_ALL_EN 99
MIN_ALL_EN 2 SB_CH 90.0 CH_EFF 0.7 SOC_AT_ARRIVAL 5 PLANNED_DEPARTURE_TIME 1449925200 ARRIVAL_TIME 1449914400 V2G 0 TARGET_SOC 72

Message from eurecat@parsec2.unicampania.it/taskmanager: DELETE_LOAD [0]:[1] 0.0 PV 47882

Message from eurecat@parsec2.unicampania.it/taskmanager: SIMULATION END

Figure 15 XMPP implementation of the dummy EMS.

This page, once the connect button is pushed, uses an http bind to connect to the XMPP server. The text fields show the current JID used by the scheduler. The xmpp communication is implemented by the **Strophe** JavaScript library. It uses the http (bosh) interface of the proxy server. The page logs the message received and the response returned.

The web page of the HTTP EMS is shown in Figure 16.

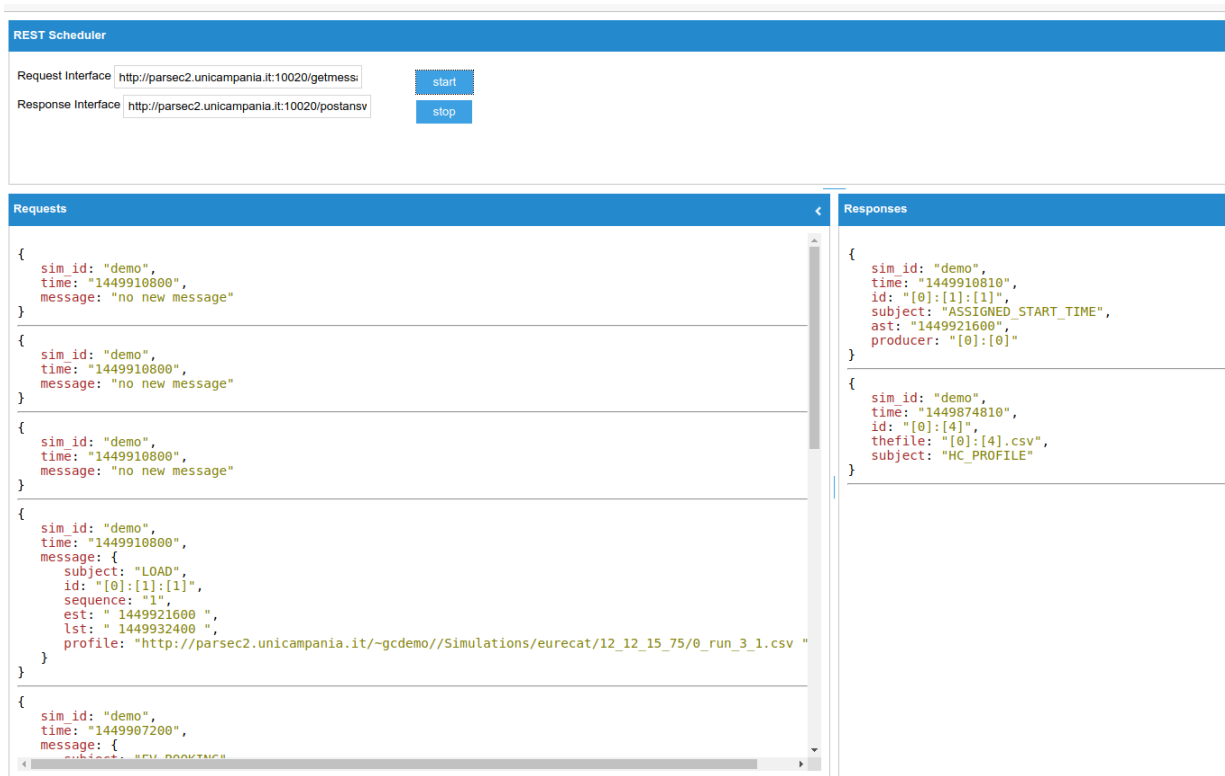


Figure 16 Rest implementation of the dummy optimizer.

This page, once the start button is pushed, periodically get message using the http interface.

The two text-fields show the urls used to get the next message (`http://[host]:[port]/getmessage`) and to post the response (`http://[host]:[port]/postanswer`).

The pages logs the message received and the response returned.

Obviously, the http communication can be monitored using the inspector tool of the browser as show in Figure 17.

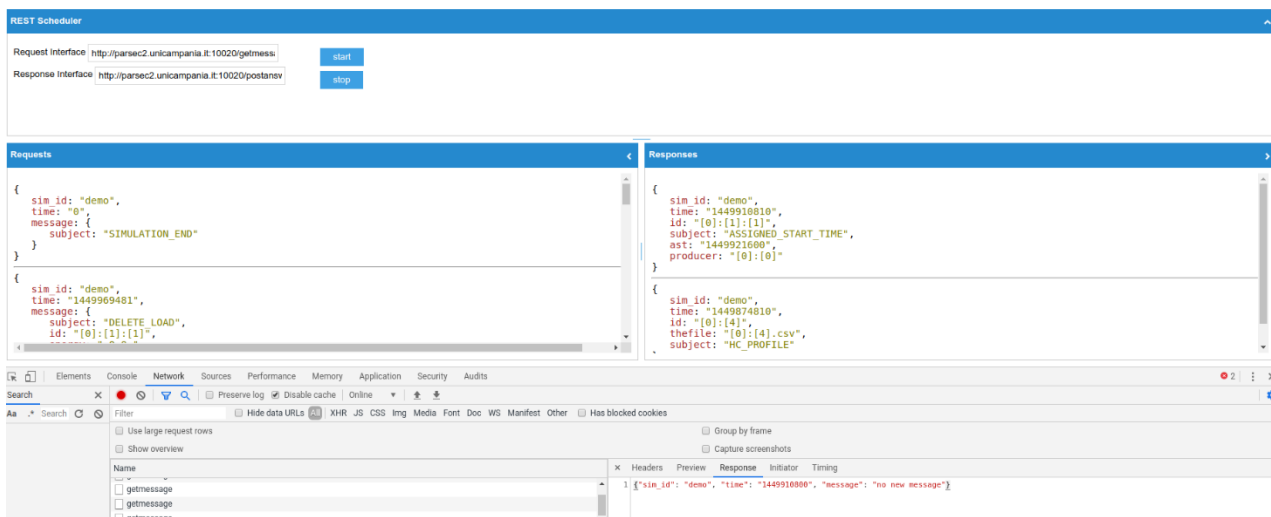
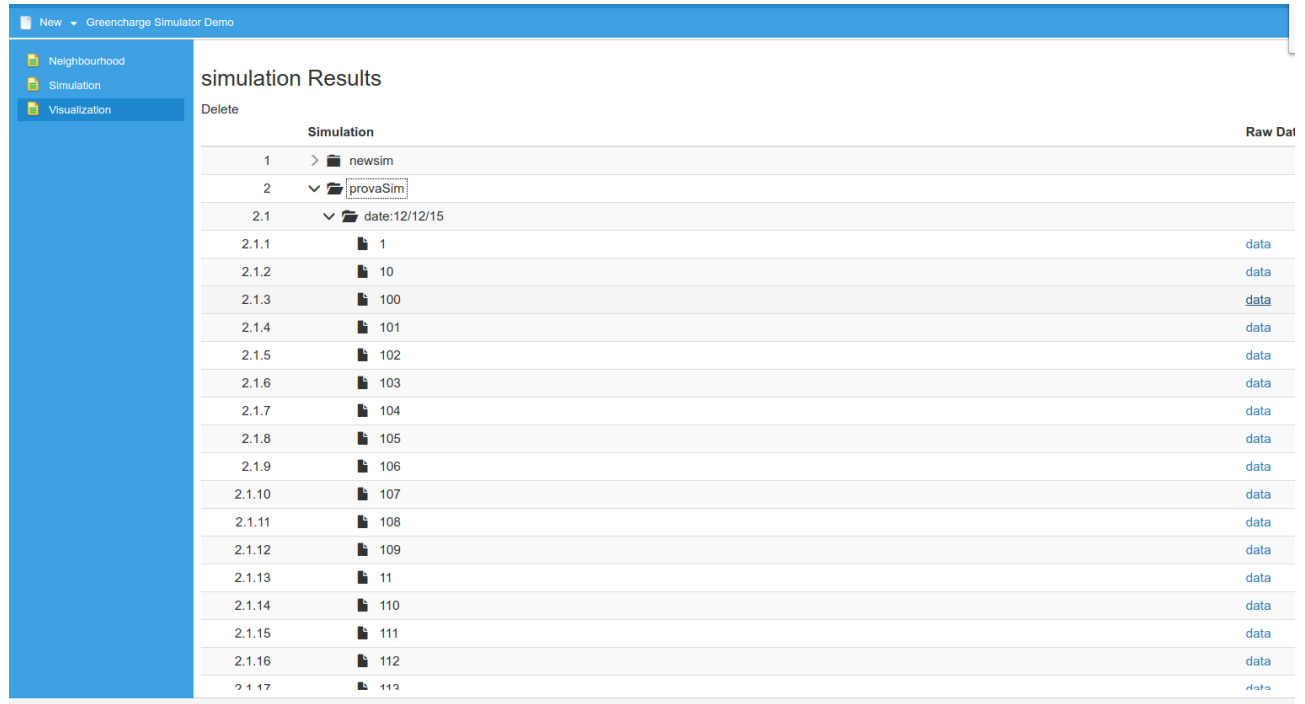


Figure 17 RAW visualization of the REST protocol.

Visualization Dashboard

The last dashboard that is shown in Figure 18 allows for the visualization of simulation results. It is possible to select the simulation run and to visualize all the input and output files generated by the simulator.



The screenshot shows the 'simulation Results' section of the Greencharge Simulator Demo. On the left is a sidebar with a 'Delete' button and a tree view containing 'Neighbourhood', 'Simulation', and 'Visualization'. The main area displays a hierarchical list of simulation runs. The top-level runs are 'newsim' and 'provaSim'. 'provaSim' is expanded to show a sub-run 'date:12/12/15', which is further expanded to show a list of 17 sub-simulations (1 to 17). Each sub-simulation has a corresponding 'data' link in the 'Raw Data' column.

simulation Results		Raw Data
1	> newsim	
2	▼ provaSim	
2.1	▼ date:12/12/15	
2.1.1	1	data
2.1.2	10	data
2.1.3	100	data
2.1.4	101	data
2.1.5	102	data
2.1.6	103	data
2.1.7	104	data
2.1.8	105	data
2.1.9	106	data
2.1.10	107	data
2.1.11	108	data
2.1.12	109	data
2.1.13	11	data
2.1.14	110	data
2.1.15	111	data
2.1.16	112	data
2.1.17	113	data

Figure 18 Tree view visualization of completed simulations.

In particular the web controls allow for opening and visualizing the csv files containing the simulation output, such as the time-series shown in Figure 19.

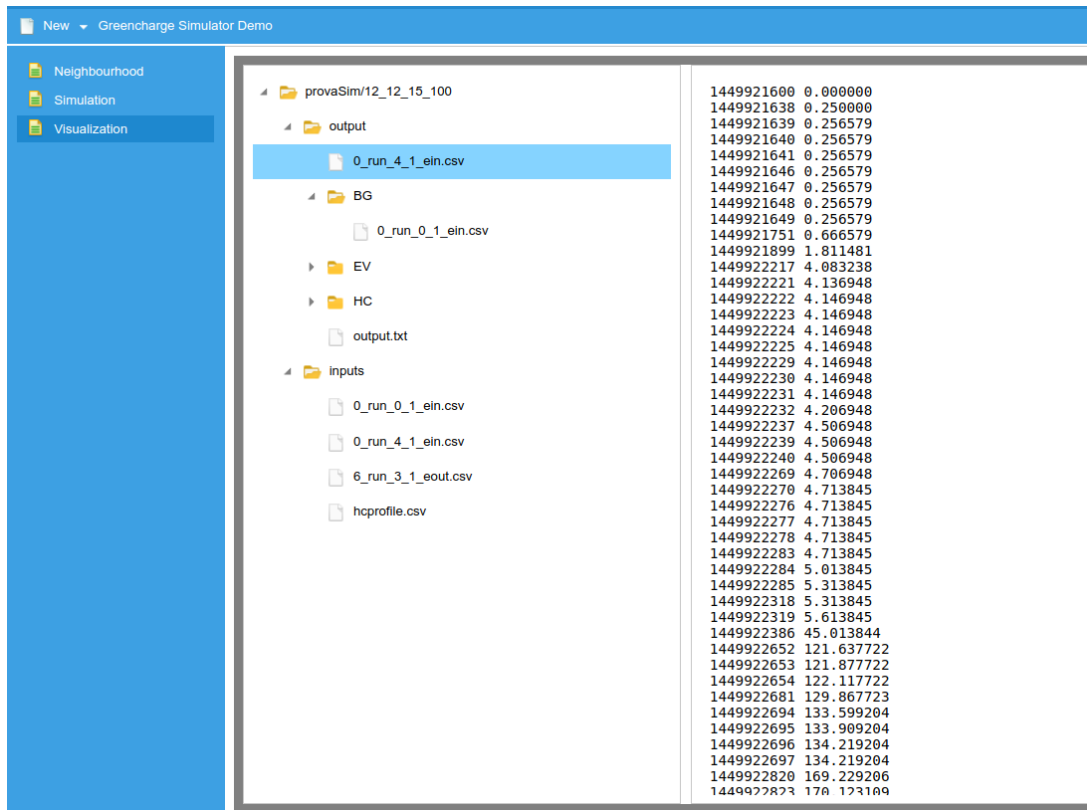


Figure 19 Visualization of simulation output.

including the list of exchanged messages for debugging purpose as shown in Figure 20.

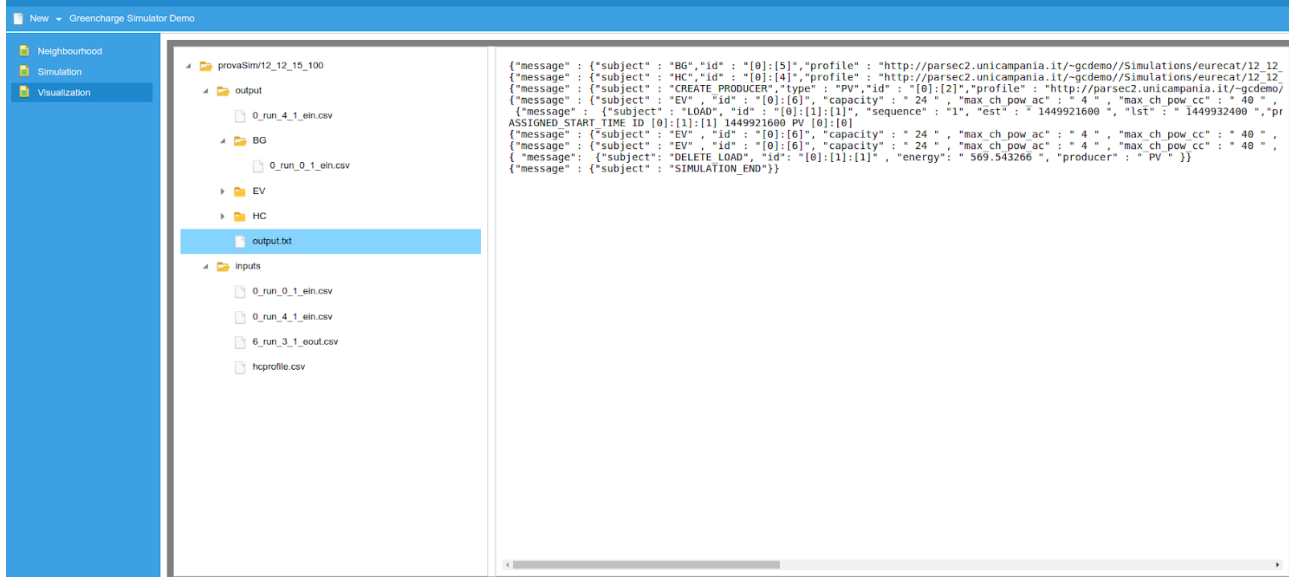


Figure 20 List of exchange messages.

4.4. UiO distributed optimiser

4.4.1. Optimisation approach

The fundamental scope of optimisation in a smart neighbourhood is to manage the energy demand to maximise the use of renewable energy. This could either be from local producer or decided by the amount of renewable

energy in the electricity mix taken from the electricity grid. The best schedule can obviously only be produced by considering the *global* optimisation problem of the entire neighbourhood. However, this may create a very large optimisation problem, and make it difficult to implement a solution on the ground as there must be one trusted agent solving the global problem for all households in the neighbourhood. This trusted agent must also be trusted by the various electricity providers for the consumers in the neighbourhood since the whole aim of the optimisation is to maximise the consumption of *locally* produced energy and minimise the consumption of energy bought from the grid electricity providers.

In practice, there is no central management of a neighbourhood, and it may be better to implement a *distributed* energy management system where each ‘restricted energy domain’ is considered an optimisation domain. The restriction is typically placed at the level of grid ‘fuses’ and so a neighbourhood domain is at the top level all consumers behind the fuse of the transformer providing grid energy to the neighbourhood. Further sub-domains will be each household with its own electricity provider and maximum fuse capacity. Finally, also separate charging stations with one or more charge points for electrical vehicles may be considered a sub-domain.

This hierarchical structure within the neighbourhood seems to suggest that distributed optimisation is possible starting from optimising the lowest level sub-domains and then schedule the electricity demand gradually by moving upwards in the hierarchy. In a practical setting, this may be easier to implement as each household may participate by implementing their own optimiser taking into consideration the local producers available to the household and the amount of renewable energy of that household’s electricity provider. The last point is important since some households may choose to pay a premium for their electricity to ensure that it comes from renewable sources, and there may consequently be differences in the ratio of renewable energy available from the grid for each household.

Given these observations it may not be a huge loss with respect to the fundamental scope of the optimisation to maximise the use of renewable energy by doing the optimisation for the sub-domains and coordinate the consumption of the sub-domains in the domains at higher levels in the hierarchy. The penalty taken by distributed optimisation over global optimisation is something that will be investigated by GreenCharge by using two different optimisers. This Section 4.4 discusses the distributed optimiser, whereas the global approach is presented in the following Section 4.5.

4.4.2. Optimiser Interface

The Network and Transport layer² connection to the XMPP server is implemented by the Swiften³ library. On top of this, the transparent communication layers of the Theron++ actor framework⁴ for C++ implements the Session and Presentation layers. The external messages are received by the session layer and mapped to the address of the internal Theron++ actor based on the actor name encoded in the XMPP Jabber identifier (JID). Outbound messages will first be serialized by the Presentation layer actor before the Session layer maps the internal actor address to the external XMPP JID.

To isolate the Optimiser’s interface with the Simulator’s event dispatcher from the other actors of the system *all* messages are exchanged with a single *Interface* actor. This actor will create the corresponding *Energy Objects* described in the following, and forward these to the *Energy group* dealing with the actual scheduling of these energy objects. The Interface actor is also responsible for timing the total time spent to calculate a new consumption schedule for the neighbourhood as measured from the time of the arrival of a message triggering a new schedule and the time the energy group for the neighbourhood reports that the schedule has been obtained. During the optimisation process various messages may be sent via the interface giving start times for consumption periods, or consumption profiles may be uploaded to the Simulator for updating the user interface.

² https://en.wikipedia.org/wiki/OSI_model

³ <https://swift.im/swiften.html>

⁴ <https://github.com/GeirHo/TheronPlusPlus>

The class structure of the interface between the Simulator and the Optimiser is shown in Figure 21.

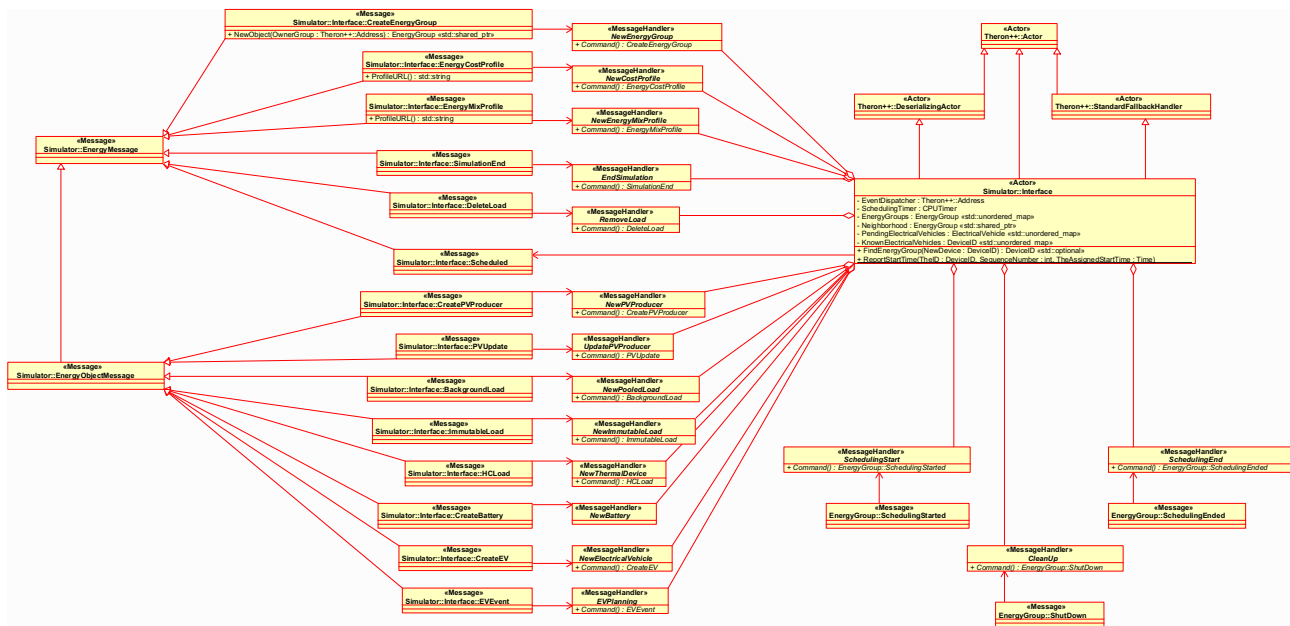


Figure 21 The interface class of the optimiser has one message handler with a corresponding message class for each of the types of messages that can be received from the Simulator's event dispatcher. The message classes are derived from common base classes supporting serialisation to and from the right XMPP message format.

4.4.3. The Energy Group

The main entity of the neighbourhood model is the *Energy group*. An energy group is an *Energy Object* that may host other energy objects and energy groups. Thus, there is one top-level energy group representing the energy smart neighbourhood and it may host other energy groups representing constrained consumption units like households or charging stations. Thus, the energy groups form a consumption tree rooted in the energy group representing the neighbourhood.

Each energy group is a *Grid* interface implementing the power constrained external connection of the energy group. An energy group may or may not host local producers of renewable energy. This leads to an immediate invariant to be respected by the simulated scenario: The sum of the grid capacity of the energy sub-groups cannot exceed the grid capacity of the hosting group. This is evident from the situation where there is no local consumption or production in the energy group and the full capacity of the group's external connection is feeding through to its sub-groups. This is the minimum requirement for the optimisation problem to be *feasible*, i.e., have a solution.

Since the identifier format defined for all energy objects are hierarchical, one can easily deduce the hosting energy group of an energy object by considering the top-level part of the identifier. This is used by the Interface object to forward directly to the right energy group each of the energy objects constructed on the basis of the corresponding messages from the Simulator's event dispatcher. It is therefore necessary that the interface actor caches the actor addresses of the energy groups it is asked to construct. The benefit of this approach is that an energy group actor will only need to know about the category of the energy object it has been assigned, and not what the energy object models. This allows the scheduler to be easily extended with new energy object models within these known categories should the current set of models need an extension.

The class structure of the energy group is shown in Figure 22.

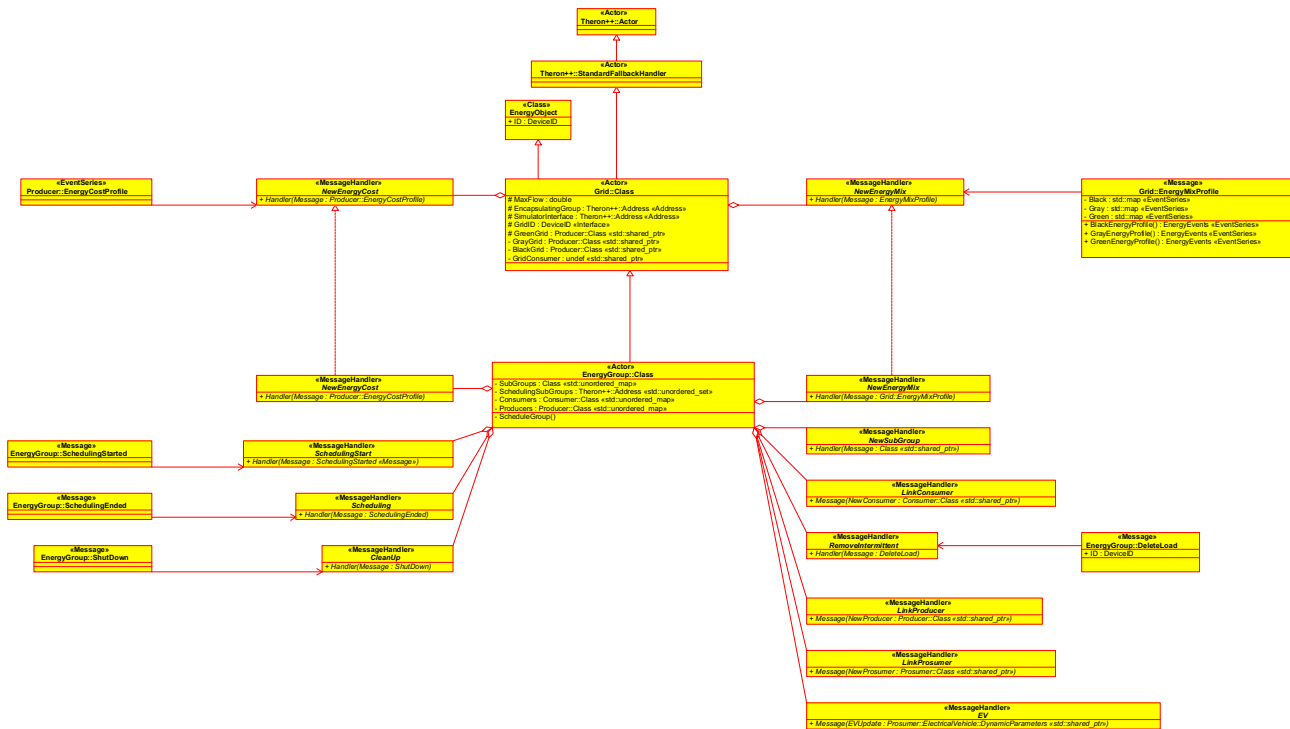


Figure 22 The energy group receives messages to create and manage categorised *energy objects*, and grid related parameters like the energy cost profile and the types of energy offered from the grid over time.

4.4.4. Consumers

Demand side energy management in energy smart neighbourhood is all about planning the consumption of energy according to the availability of renewable energy, or the availability of green energy from the electricity grid if the grid energy mix is known. Consumers are therefore an essential category of *Energy objects*. In time boxed discrete scheduling, each consumer will have a *fixed* power demand per time box, i.e., consumption interval. For *inflexible* consumers, this power demand is a single scalar, but it can also be an interval indicating that the consumer can be *flexible* and should receive power in this interval.

The first type of consumer is the background load defining a *persistent* consumer that must have the required power in all time boxes. Washing machines and dishwashers are examples of *immutable* consumers. These are flexible regarding the time to start the consumption, but once started the device should run until completion. Finally, there are heating or cooling devices that has a nominal consumption profile but can tolerate deviations from this profile provided that the deviations are recovered timely. Figure 23 shows the class structure for the consumers.

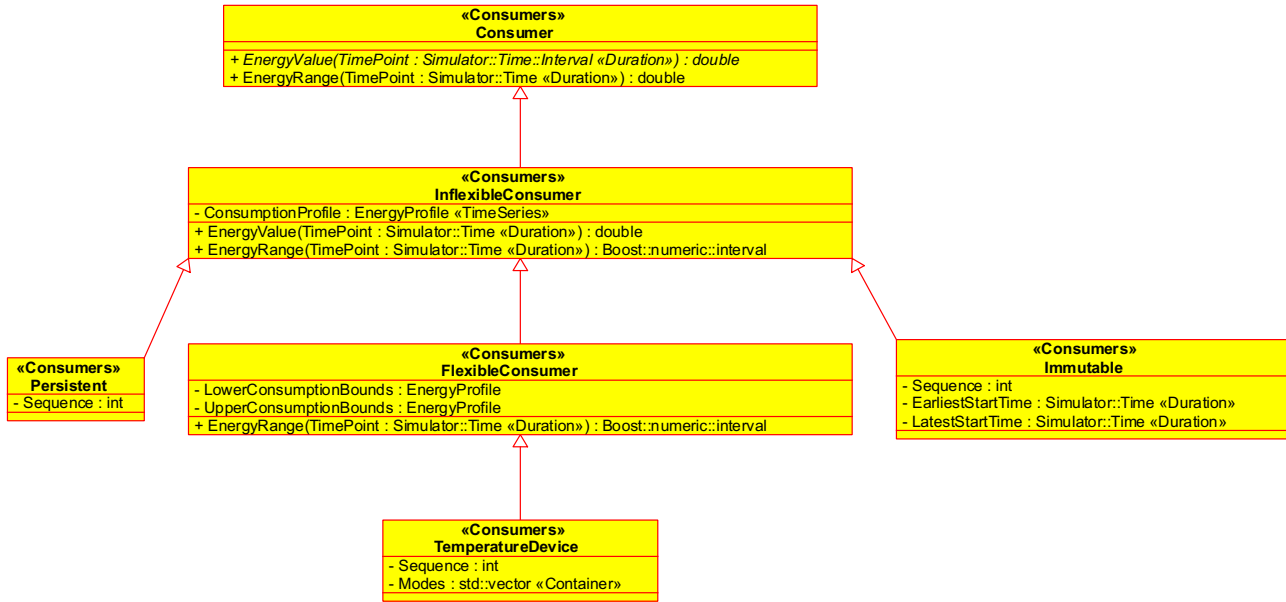


Figure 23 The consumer types of the distributed optimisers

4.4.5. Producers

The producers are characterised by having a *production* profile predicting how much energy the producer has produced until a given time, i.e., the cumulative production profile. This prediction is updated regularly based on new weather forecasts since small scale renewable energy production largely depends on the weather. A photo voltaic panel is thus an example of a *variable power producer*. On the other hand, a producer may also be a battery that has been charged when available renewable energy exceeded local demand and can discharge at *constant power* when there is little or no local production. The class structure of the producers is depicted in Figure 24.

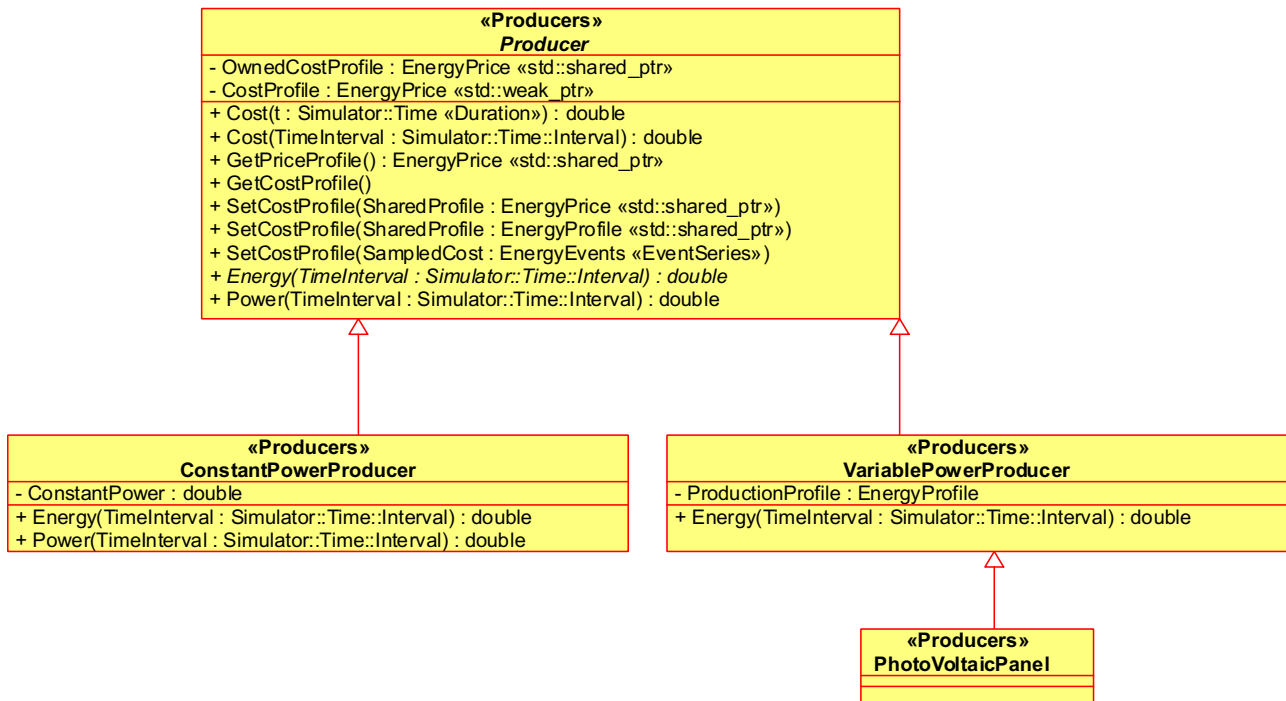


Figure 24 The producer types of the distributed scheduler

4.4.6. Prosumers

A prosumer is an energy object that may consume energy in certain periods and deliver energy to the consumers in the neighbourhood in other periods. A battery is the obvious example of this category. It may be a stationary battery or a *dynamic battery* that supports charging and discharging only at given periods. An example of the latter is a mobile battery, i.e., an electrical vehicle, that may or may not support delivering energy back to the local consumers. Figure 25 shows the structure of the producer classes of the distributed optimiser.

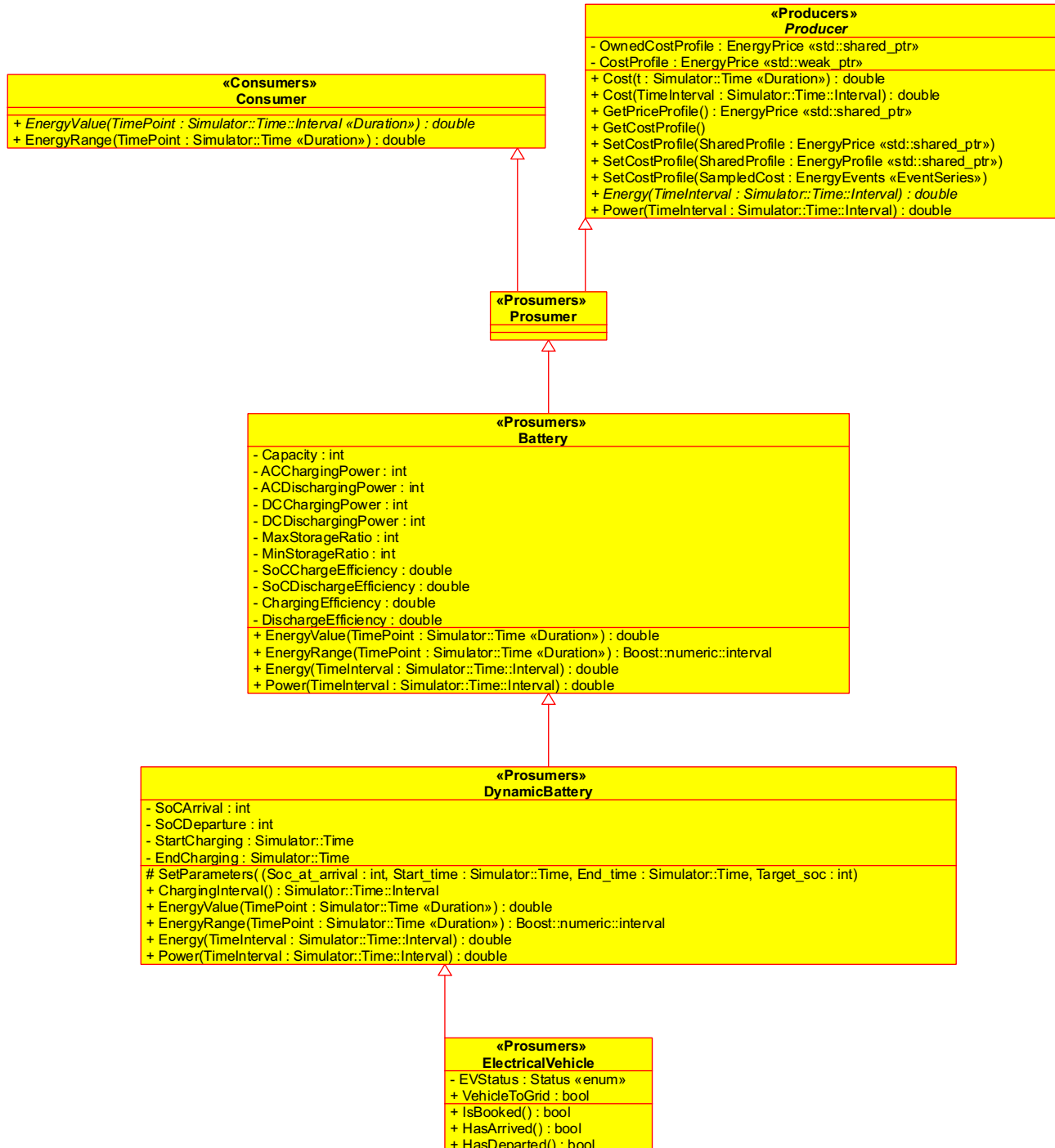


Figure 25 The *prosumers* that are able to act as both a consumer and a producer of energy in a smart neighbourhood as implemented in the distributed optimiser.

4.5. EURECAT Optimiser

4.5.1. Optimiser approach

As presented previously, there are two different approaches: local (or hierarchical) and global. The Eurecat optimiser follows a global optimization approach. All the elements in the scenario are optimized with a single goal function. That is, the best solution is the one that minimizes the total cost of the energy and the carbon footprint to fulfil the energy demand of loads and electric vehicles. It is assumed that the owners of the different devices act as members of an energy community and trust a third party to operate it. The solution obtained in the global optimization is better (or equal in worst case) as the solution obtained by the addition of individual optimizations (hierarchical). By contrast, the scalability is a drawback of this approach. As the number of devices grows, so does the number of variables involved in the optimization problem. Although there are very efficient algorithms to solve big problems, the increasing number of variables may be an issue. In such a case, there are two alternatives:

- Decrease granularity: The period to optimized is discretized. That is, the plan provided by the optimised will only provide values for certain moments in time, typically, with the same granularity as the energy values are sampled. Then, the number of variables the optimiser must calculate is proportional to the sample ratio. Thus, if instead of finding the best plan with a granularity of 5 minutes, the optimiser provides commands (set points) that happens every 15 minutes, it will roughly decrease the number of variables by 3.
- Group devices: For some devices, a supermodel can be created that encapsulates the behaviour of individual entities. This strategy may resemble a hierarchical approach, but still keeps the goal to perform a global optimization. An example of such grouping can be the object charging station: the arrivals and departures times and the energy demand of each individual vehicle can be translated into energy flexibility as described in section 3.1.4. The optimiser calculates the best plan for the charging station, and then, a new optimization has to be performed to calculate the best set point for each individual vehicle.

Another particularity of the optimization approach followed is that everything has a cost assigned and constraints can be seen as hard and soft constraints. For example, a power limitation may be a physical constraint (hard constraint) or a soft constraint (coming from a Demand Response Request). The cost (or penalty) assigned to exceeding the overlimit will modulate the intensity of the constraint. The advantage of this approach is that even if it is not possible to find a solution that fulfil all the limitations, the optimiser will provide a solution and it will be possible to analyse why it was not feasible (when and how much the limit was exceeded).

4.5.2. Optimiser Interface

The optimiser interacts with the Simulator Dispatcher through an http interface. On the site of the Optimiser, a Django module runs continuously in a docker. This module implements two models: one to save all received GET calls as objects and the other to save all responses sent as POST.

The optimiser controller is a class called `integrationThread`, this class has three main functions:

- `getCall`: currently calls (<http://<url>/getmessage>). It queries every second and saves the response to the database after doing the necessary checks.
- `calculatePost`: calculates the response to be sent. (only calculates the response if the subject of the received GET message is 'LOAD', 'HC' or 'EV').
- `postCall`: in case there is a response generated from the above function, `postCall` makes a post (<http://>url>/postanswer>)

Errors control:

getCall and postCall take care of preparing the next call before executing. If there is an error, it is collected and saved in the database and the class integrationThread continues working independently.

The mapping between the objects in the simulation scenarios and the internal model of the optimiser is shown in Figure 26.

The internal model has loads, generators, batteries, EVs and Grids. Loads can be identified as consumers, generators can be identified as producers, but the classification of prosumers is somehow more complex. Instead, the internal model defines EVs that have the same behaviour as batteries, they can be charged or discharged (V2G), but they additionally have an arrival time and a departure time. The object grid is not considered as a generator since it allows bidirectional flow (import and export of energy).

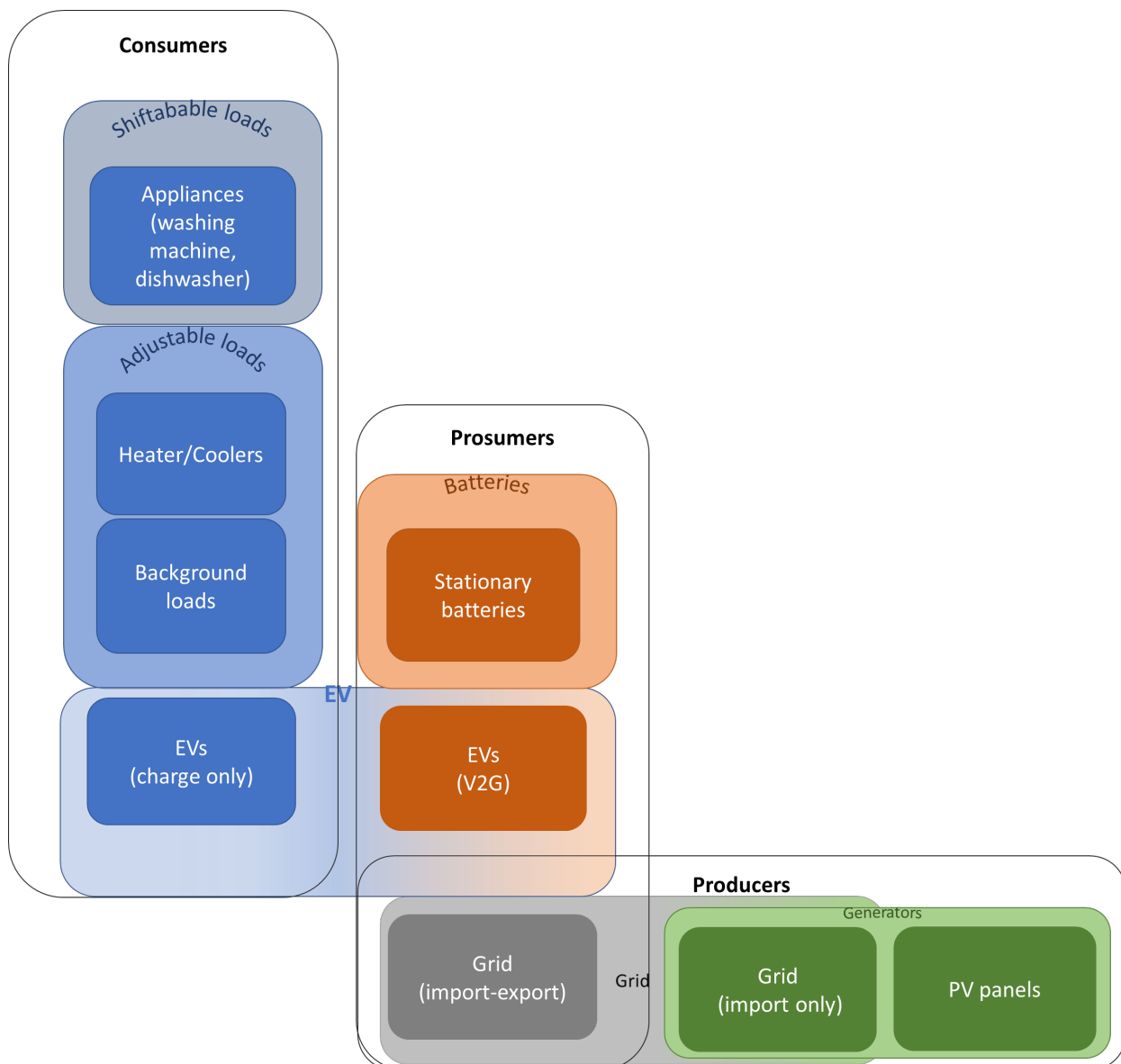


Figure 26 Mapping between the optimiser and the simulator objects

4.5.3. Optimiser engine

The core of the optimiser is a module called Calculator.

Algorithm:

To find the best solution (according to a goal function), it implements a combination of algorithms: simulated annealing and linear programming. While the simulated annealing is used as a search algorithm to find the best combination of starting times for the shiftable loads, the linear programming takes advantages of the linear behaviour of the batteries for EVs and stationary batteries. Adjustable loads are also integrated in the linear programming.

Goal function:

The goal function comprises the following factors:

Cost of energy: The cost of energy is calculated taking into account the cost of the energy used according to the source of origin (producers and grid). The producers and the grid have an assigned cost that may vary with a certain resolution (the maximum resolution is a configuration parameter).

Carbon footprint: Similarly, to energy cost, the carbon footprint is computed by taking into account the energy consumed according to the source of origin. Each source has assigned a carbon grid intensity defined by the production technology (i.e: PV). In the case of the energy coming from the grid, the grid intensity is calculated using the energy mix composition, that varies along time.

RES waste penalty: Additionally to compute the carbon footprint, an extra factor controls the relevance of consuming the energy produced locally. In this sense, a factor penalised the energy of local producers that cannot be consumed. This factor can be used for scenarios where it is not possible to inject surplus of energy to the grid or where the energy coming from the grid is as green as the energy locally produced.

Delay penalty: This factor was introduced to break ties, in the sense that when 2 different solutions (or plans) had the same associated cost, an extra cost will be added to penalise the solution that satisfies the energy consumption later, even if it is before the time constraints defined. In general, it was meant to be a very tiny cost. In practice, it will be applied to reproduce the behaviour of a greedy scheduler or a base line scheduler, meaning an optimiser that just takes into the account technical constraints.

Overlimit penalty: It is a factor that is useful in cases where the power limits are not physical or hard constraints, but they can be exceed although it is not desirable. This is the case of demand response requests, for instance. In case of hard limits, this penalty will be extremely high.

All these factors are weighted, and these weights are configurable.

4.5.4. Loads

The consumers are represented by the object Load of the optimiser. There are two types of loads:

Shiftable loads (programmable loads): The typical examples are washing machines, dishwashers and alike. The load curve is fixed but the activation time can vary within a time period.

Adjustable loads: There are loads which allows a certain modulation of the power consumed. Despite the shiftable loads, they cannot be activated or deactivated. They allow to define different ranges with corresponding increasing costs. That is, the cost to reduce the power a certain amount depends on the amount. For instance, reducing 100 W has a cost 2, while reducing the power 200 W has a cost 5. A reduction of power higher than 200 W, in the example, will have a very high cost (infinite) expressing that it is not a good solution for the optimiser. Background loads are modelled as adjustable loads with no adjustable range.

All devices, including loads, are modelled with a section describing the technical parameters and a section describing the so called economic parameters, that reflects penalties or costs. As an example, a sample for the shiftable load data model is shown in Figure 27:

int	program_length	Number of data points.
time_t	t [MAX_PROGRAMLEN]	Data points: Time (relative to the activation) [s].
double	p [MAX_PROGRAMLEN]	Data points: Active power [kW].
time_t	startt [MAX_BREAKPTS]	Available start times.
int	nstartt	Number of available start times in startt .
double	penalty_off	Penalty for no service.
double	penalty_sec	Penalty for each second of delay.

Figure 27 Internal representation of shiftable loads in Eurecat optimiser

4.5.5. Electric Vehicles

Electric vehicles are handled as a separate object different from loads and batteries. The reason is that although the behaviour is similar to a battery, their presence is discontinuous, they arrive and go. They are not modelled as a load either, because although they behave as a consumer that is demanding a certain amount of energy within a time period, they can also behave as a producer if a configuration of V2G is allowed.

The technical model of the vehicle (battery) is described through a linear model and a maximum charge and discharge power. It is considered that the power can take any value within this range. In practice, most charging points only allow switching on and off the power transfer. In those cases, an additional step is taking to convert power variation into on/off signals. The internal representation of an electric vehicles is shown in Figure 28

time_t	timeOn	Start time of the relevant time interval.
time_t	timeOff	End time of the relevant time interval.
double	initLevel	Initial energy level [kWh].
double	finalLevel	Demanded final energy level [kWh].
double	maxLevel	Maximum energy level [kWh].
double	maxChSpeed	Charging speed [kW].
int	v2g_capable	Has v2g capacity?
double	penalty	Priority (penalty for not arriving to the demanded level) [e/kWh].
double	bonus	Bonus for exceeding the demanded level [e/kWh].
double	ChSpeed	Current charging speed.

Figure 28 Internal representation of electric vehicles in Eurecat optimiser

4.5.6. Generators

The generators handled in the optimiser are renewable generators. In particular, the models available are for mini wind turbines and PV panels. The model to be used in the simulator are PV panels and the modelling of the panel is handled directly by the forecasted production. The economic parameter included in the model are the price of energy generated, as a constant value.

4.5.7. Grids

The object Grid models the interconnection of the neighbourhood with the global grid. There can be more than one interconnection, meaning, more than one object Grid. It is not model as a generator because the interconnection allows bidirectional flow, that is the neighbourhood can import energy from the grid when the consumers demand more energy than the one that can be supplied by local generators and batteries, or it can export energy to the grid when there is a surplus of energy generated locally that cannot be stored or consumed or the feed-in tariff is attractive enough to sell it to the grid.

4.5.8. Batteries

Similarly to EVs, the batteries are assumed to behave linearly. This approach is a good approach when the state of charge range between 20% to 80% of the battery capacity. The batteries can be charged or discharged and an efficient coefficient models the losses. State of charge at the beginning and end of optimization are defined as boundary conditions (boundCond). The constraints expressed in TConstraints (Figure 29) are populated taking into account the maximum power (maximum charging/discharging speed) and the total capacity of the battery.

int	status
char	id [MAXPIDLEN]
double	SOC Current state of charge [kWh].
TPower	setPoint
TList *	boundCond
double	maxLevel Maximum energy level [kWh].
double	penalty Priority (penalty for not arriving to the demanded level) [e/kWh].
double	bonus Bonus for exceeding the demanded level [e/kWh].
TConstraints *	constraints

Figure 29 Internal representation of a battery

4.5.9. Neighbourhood

There is not a formal object to define the neighbourhood in the optimiser. The way it has been chosen to model it is through constraints, representing the maximum amount of energy (or power) that can flow through a link (or interconnection). For instance, a charging station is a collection of charging points and the maximum energy that can be delivered is constraint by the fuse of the charging station.

5. The KPI Calculator and Visualization Tools

The KPI Calculator and Visualization Tools can be reached at:

<http://parsec2.unicampania.it/~branco/gccalculator/>

The related guidelines have been published at:

<http://parsec2.unicampania.it/~branco/gccalculator/calculatorUserGuide.pdf>

The calculation and visualization software of the KPIs is divided into two distinct blocks that interact via REST API:

- GreenCharge Calculator
- GreenCharge Visualization -Tool

The design choice to divide the two components derives from the decision to want to decouple the calculation part from the visualization part in order to offer the end user also software that is capable of performing the calculation via bash.

The main components are located on two different servers. The first is responsible for keeping the data and the second for processing it. The temporary archive on the "Web Server" will contain the data useful for calculating the KPIs only in the period strictly necessary for the calculation, they will be deleted immediately afterwards. Only the results of the calculation will be saved in different files sorted by user with a directory structure that will be discussed afterwards.

5.1. The GreenCharge KPI Calculator

List of KPIs already calculated:

The number of EVs in general.
The number of charging point (CPs) are already available for charging
The number of CPs are private.
The number of CPs are shared.
The time EVs are connected during a specific time span
The time the EVs are charging compared to the total connected time
The energy EVs are charged with per connected time unit
Number of charging sessions during a specific time span
Energy transferred to the EV battery compared to energy demand
Share of charging sessions where EVs are charged according to the charging demand
Share of booked time slots that are not used
Average delay in plug-in time
Share of sessions that are not finished (plug out) in time
Average delay in plug-out time
Share of battery capacity for V2G
Flexibility provided by user in charging case
Actual flexibility
Flexibility provided by user in V2G case
Peak to average ratio

Self-consumption

List of KPIs Designed and Graphically developed but not fully implemented:

Average operating costs
Average personnel costs
Average energy costs paid to RES producer
Average energy costs paid to DSO/TSO
Average maintenance costs
Extra energy costs from measure paid to RES producer
Extra energy costs from measure paid to DSO/TSO
Total Capital Investment costs
Average operating revenues in general
Average operating revenues linked to energy costs/prices
Average total cost per kWh in a period
Average cost linked to energy use in a period
Average cost per kWh in a period linked to a specific price list

5.2. Overall Design of the Calculator

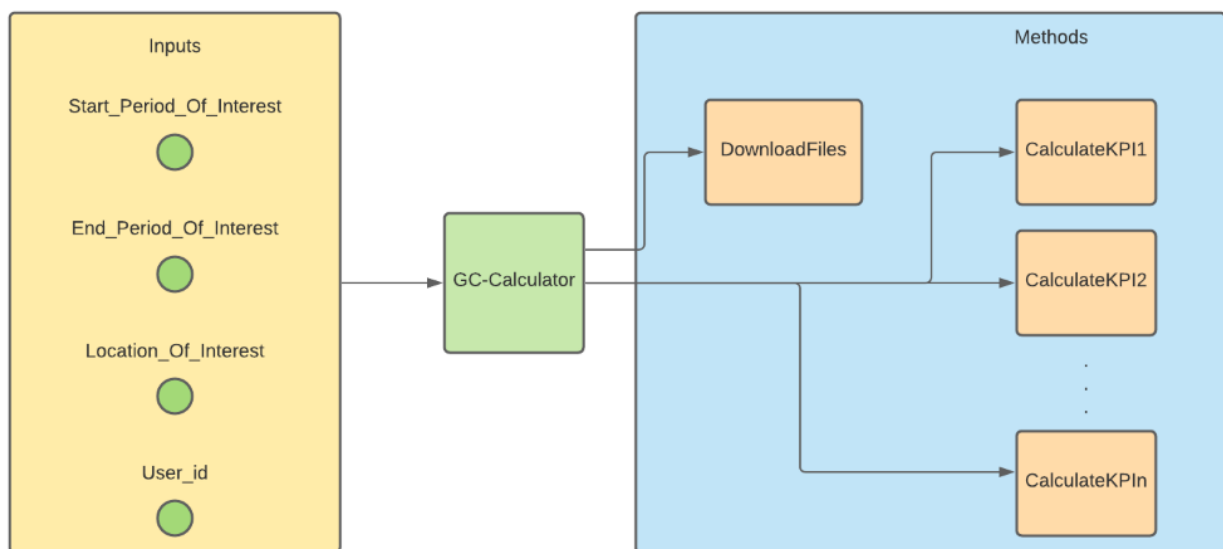


Figure 30 GC-Calculator Scheme

Given the simple structure of the problem, which consists in the calculation of indicators starting from a set of input files that contain the data useful for the computation, it was decided to use a batch procedure to achieve the goal. Therefore, the calculator has no classes inside, it simply takes in input a set of parameters (represented in Figure 30) Which :

- Start_Period_Of_Interest
- End_Period_Of_Interest
- Location_Of_Interest
- User_id

downloads the files of interest from the SFTP server and sequentially calls the calculation functions of the various KPIs. Each function saves the results in a csv-type associative file in the user's folder corresponding to the user_id input to the various KPI calculation functions.

Calculation phases:

- The user uses the dashboard to select the input parameters, the dashboard starts the KPI calculation software via a REST interface.
- The calculation software tries to connect to the Sftp server and returns an error in case of problems.
- The script tries to download only the files that are relevant for that particular calculation session. If there are no relevant files, an error is returned.
- The data is extracted into data structures of the "Key" - "Value" type, where the "key" is the name of the attribute present in the downloaded file and the "value" is the corresponding value within the file.
- The timeseries, on the other hand, are saved in a three-dimensional matrix where the i-th entry represents the timeseries while the j-th represents the time value and the k-th represents the measured value.
- After extracting the data, the actual calculation is performed using the algorithms defined for each individual KPI.
- At the end of the calculation all the used files are eliminated and only the results are saved in files.
- Finally, the dashboard takes care of reading these newly created files and showing them to the use.

From an implementation point of view, the GreenCharge Calculator was implemented in Python3 and uses some libraries that allow the connection to SFTP servers in order to download the needed files for a given calculation and to enable users to read and manipulate Excel Files of Business KPIs.

In particular:

- openpyxl is a Python library to read/write Excel 2010 xlsx/xlsm/xltx/xltm files.
- Python ftplib is a module that implements the client side of the FTP protocol. It contains an FTP client class and some helper functions.

Given its nature, the entire body of the software was designed as a sequential application that calls a specific function for each goal.

5.3. Data Ingestion and Curation

The Datachecker deals with the verification of the data loaded in the shared repository and its task is to ensure compliance with the agreed format for the data. The check is performed both on the file name and on the data it contains since both have, for each type of data, a different format to follow. Furthermore, the control is able to notify the data providers via eMail when the software finds any inconsistency in the files added or modified. The software also provides a graphical interface that helps data providers and managers to have an immediate overview of the amount of data loaded into the repository and how many of them are correct or incorrect, it also provides information and statistics on how much data has been loaded for each type, how many are correct and how many are incorrect for each of the data providers, proving to be a very useful tool for the population of the data lake. The tool also provides precise indications to the data providers regarding which checks have not been respected and which ones have been carried out and performed successfully for the data that are not correct.

The Datachecker software has been implemented in python with a supporting web interface developed in php and Bootstrap. The software runs in batches and is divided into abstract classes that define the three types of input data with implementations of the abstract classes that are different for each type of data. Finally, each class has methods that perform the various checks on the data structure and on the values they contain. The work of Datachecker is indispensable for the correct functioning of the other component of the pipeline that deals with the analytics, since a correct syntax, structure and formation of the data is necessary for its correct functioning. We provide some screenshots of the web interface that exposes the Datachecker on the analytics performed regarding data curation in Figure 31 and Figure 32 that is useful for data providers who have a web interface at their disposal that informs them of any problems in the data loaded into the storage and offers a timely report in case of data errors.



Figure 31 DataChecker General View



Figure 32 DataChecker Detailed View

5.4. The Visualization tool

In this section we will discuss the KPI viewer.

The calculator is activated, to ensure integration with the visualization tool, with a server that exposes REST calls.

When a user requests to start an evaluation session, the web page sends a REST request to a web agent that runs the actual GCCalculator and waits for the calculation procedure to finish. The GCVisualizer at this point polls another REST call, this call returns true if the KPI computer is still doing its work and returns false if it has finished. The viewer shows a loading page until the calculator has finished its work, after which the GCVisualizer shows the results of the calculated KPI.

The GreenCharge calculation and visualization tool aims to allow users to view KPIs via a web interface. The interface has been designed to be as simple as possible but which simultaneously offers all the necessary functionality.

The KPI visualization tool mainly concern the visualization of three categories of KPIs.

- In the first category we have the businesses KPIs for which we need a time period ranging from one year to three years (the calculation of the single value is carried out month by month).
- The second category of KPI includes a single KPI: "POWER PEAK KPI". For the calculation of this KPI it's necessary to specify the component of the neighbourhood together with the time period of interest.
- The third and last category includes all the other KPIs that requires to specify the time interval and the Pilot site.

As for the authentication mechanism, sequence diagrams are provided that show how it works, with a screenshot of the login page where the user must enter his credentials before making an evaluation. The login process is essential for the correct functioning of the software as it needs to know the id of the user who wants to carry out an evaluation session. The reason for this choice is the division into folders of the output results. In fact, at each session, the GCCalculator saves the results in the directory corresponding to the user who started it and the viewer reads the data saved from the same directory.

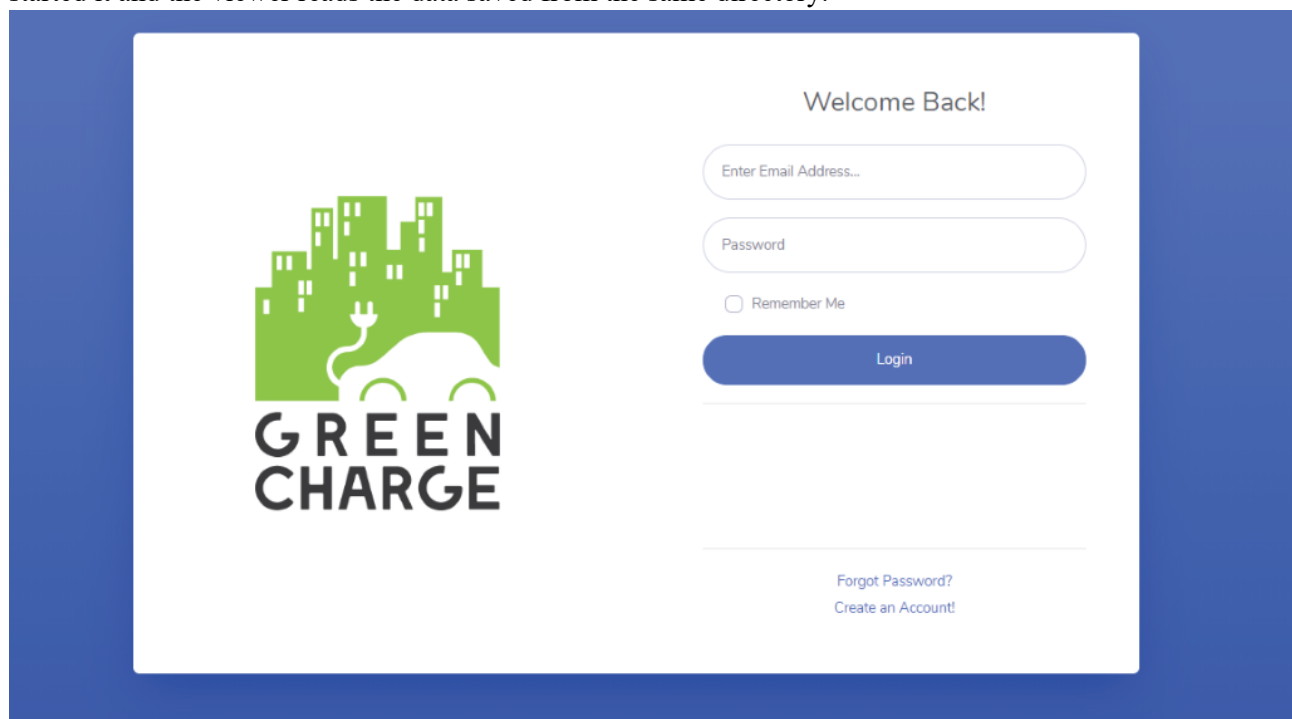


Figure 33 Login Page

Once logged in, you will be redirected to the following main page (Figure 34). In this page you can view a group of KPIs that are in numeric or percentage form. This representation is effective because it allows the user to have most of the numeric value KPIs in a single screen immediately after the start of an evaluation session.

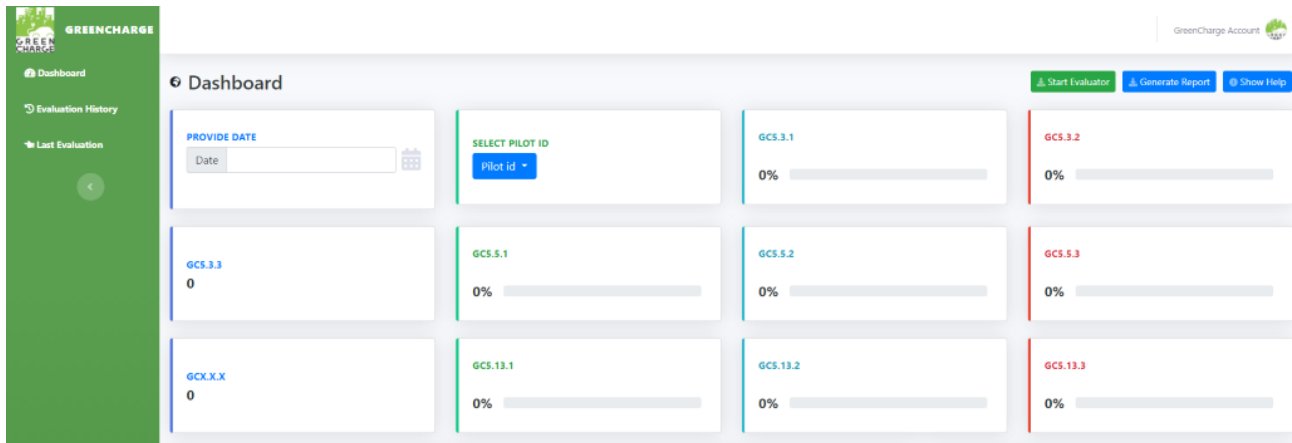


Figure 34 Visualization Tool Top View

Before proceeding with how to start an evaluation session, two features of the visualization tool are shown that may be useful to the user. The first is the KPI legend, which can be viewed using the "Show Help" button. This legend is useful since in the dashboard the KPIs are listed by their own code, the legend maps the code with the description of the KPI making it much easier to read the data. The legend will appear on the left side of the page.

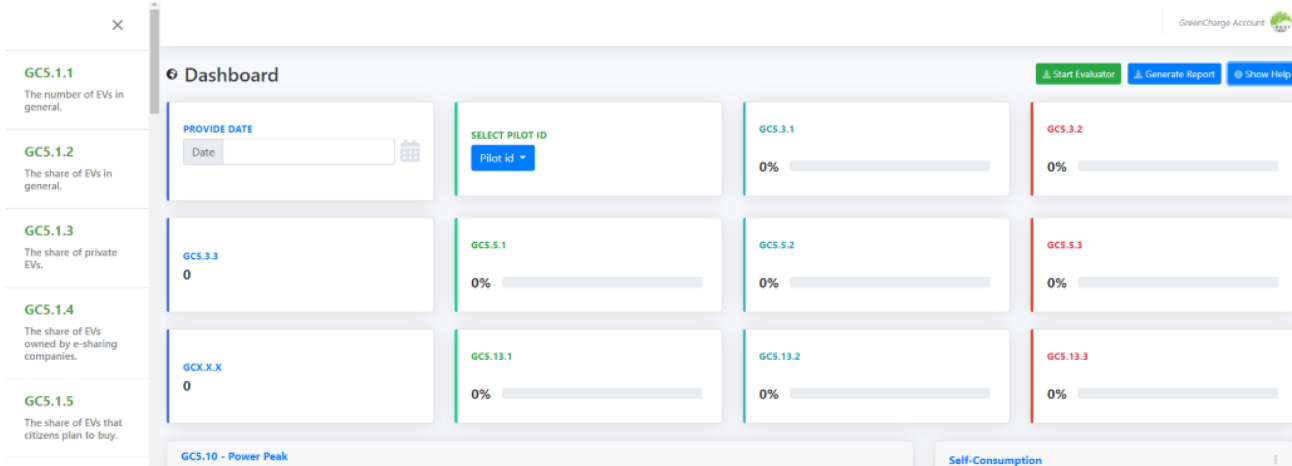


Figure 35 Visualization Tool Legend Help Bar

Another feature is that of the history of the evaluations performed, accessible via the menu on the left of the page, and it shows for each evaluation when it was performed and the parameters given in input.

History				
PROVIDE DATE				
#	Start Day	Last Day	Pilot ID	Evaluation Date
13	1 February 2020	2 February 2020	1	2020-10-23 04:15:16
14	1 October 2020	2 October 2020	1	2020-10-23 10:00:55
15	1 October 2020	2 October 2020	1	2020-10-23 10:01:01
16	1 October 2020	2 October 2020	1	2020-10-23 10:03:16
17	1 October 2020	2 October 2020	1	2020-10-23 10:03:18
18	1 October 2020	2 October 2020	1	2020-10-23 10:03:21
19	2 October 2020	3 October 2020	1	2020-10-26 02:18:09
20	2 October 2020	3 October 2020	1	2020-10-26 02:19:20

Figure 36 Evaluation History Table

To start a new evaluation session, the user must select the period of interest from the drop-down menu that opens in the "provide date" TAB. Then select a location ID in the drop-down menu immediately to the right of the previous one.

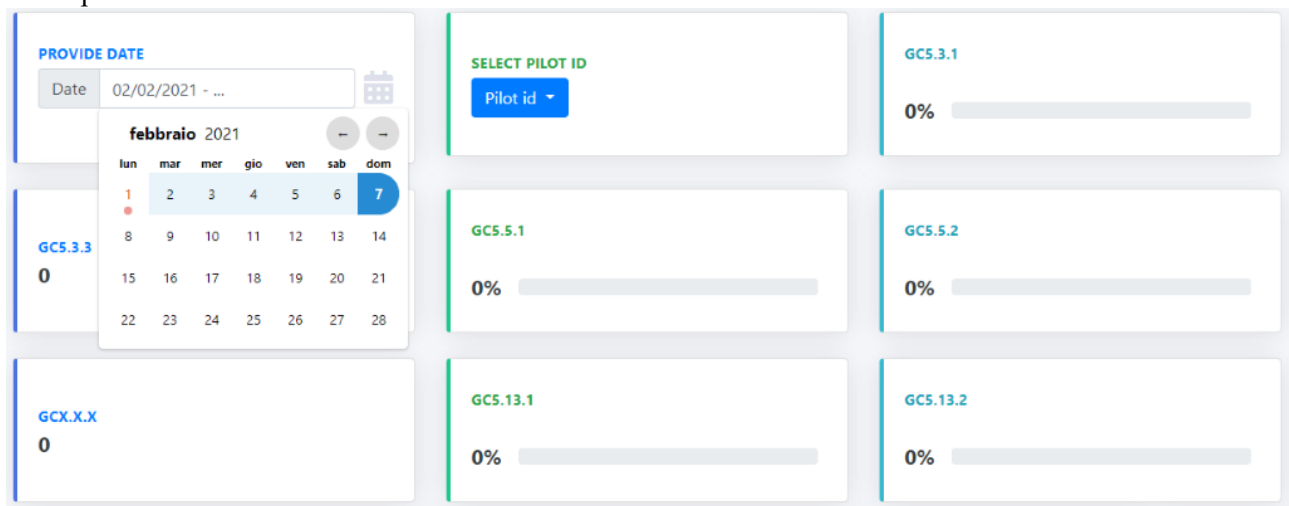


Figure 37 Date Selection

Once these operations have been performed, click on the green button at the top right: "Start Evaluation", a loading screen will appear as in Figure 38. The loading time may vary according to the amount of files that the GCCalculator will have to download to carry out an evaluation session, logically, the larger the period of interest, the greater the number of files that are useful for calculating the KPIs, so the waiting time can vary and may even be a few minutes. At the end of the calculation the web page will automatically redirect the user to the main page which will show the results.

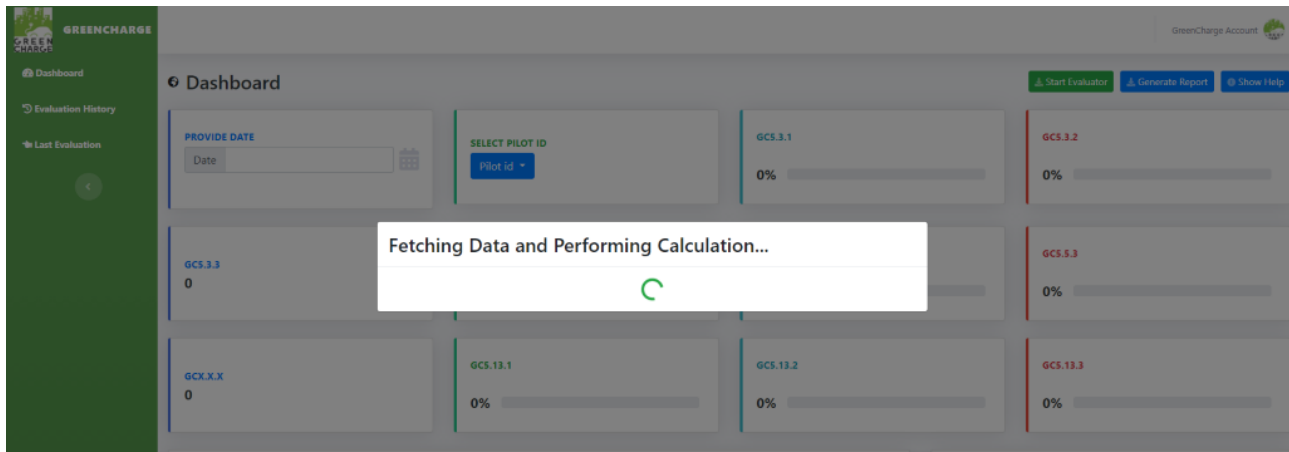


Figure 38 Loading Screen

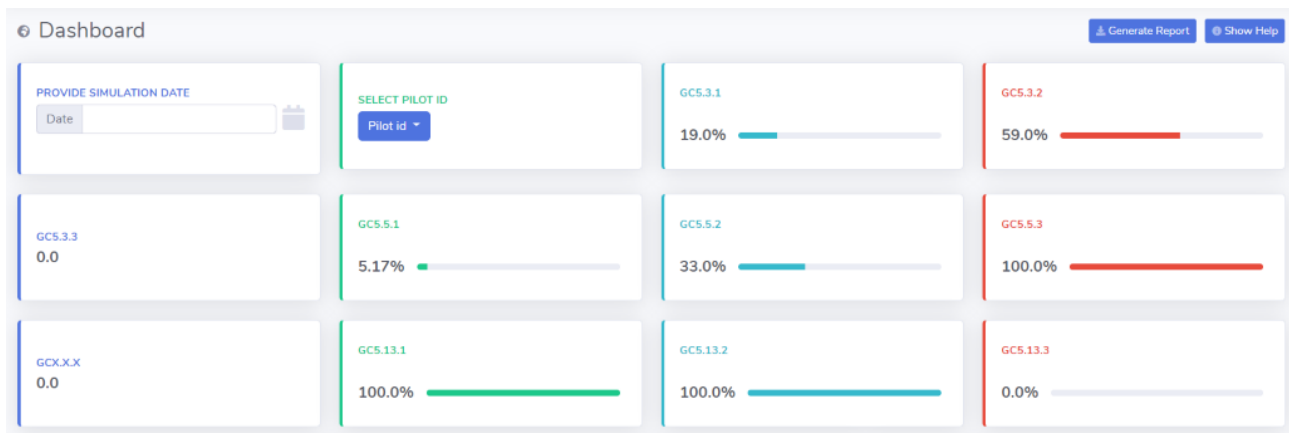


Figure 39 Example of Visualization of Some KPIs

The first group of KPIs (which are numerical or percentage type) will be shown as in Figure 39. Furthermore, the graph of the self-consumption will be shown with the relative total percentage value over the period of time selected as in Figure 40 .

The graph is able also to show total production, total consumption, and consumption from the grid.

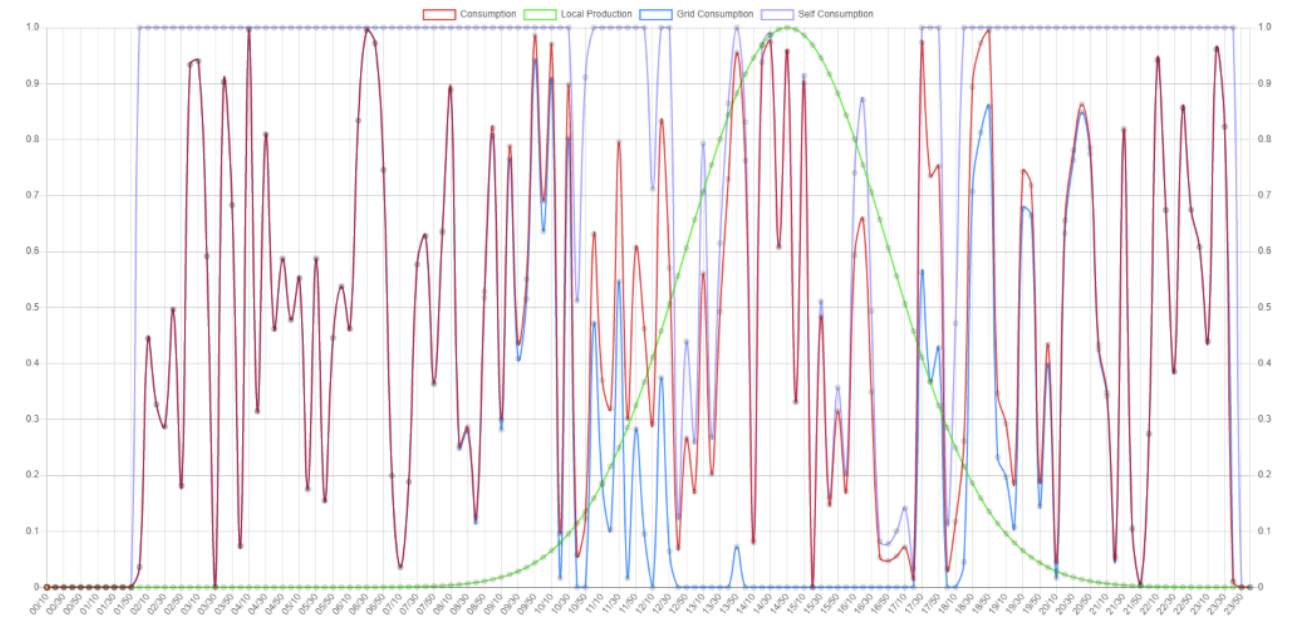


Figure 40 Self Consumption Chart

Finally, it is possible to view the KPI businesses by selecting the pilot, the demo site and the time period in years (2019 -2021). It's possible to hide series simply by clicking the KPI code on the legend and download the graphics in PDF, JPEG or SVG format.

As regards the KPI GC5.10 - Power Peak, since the input data are different, it was necessary to create other dropdown menus that give the necessary data as input to the viewer (this is due to the fact that the KPI in question can be calculated for the whole demonstrator, for the whole pilot site, for a single charging station or even for a single charging point as you can see in Figure 41).

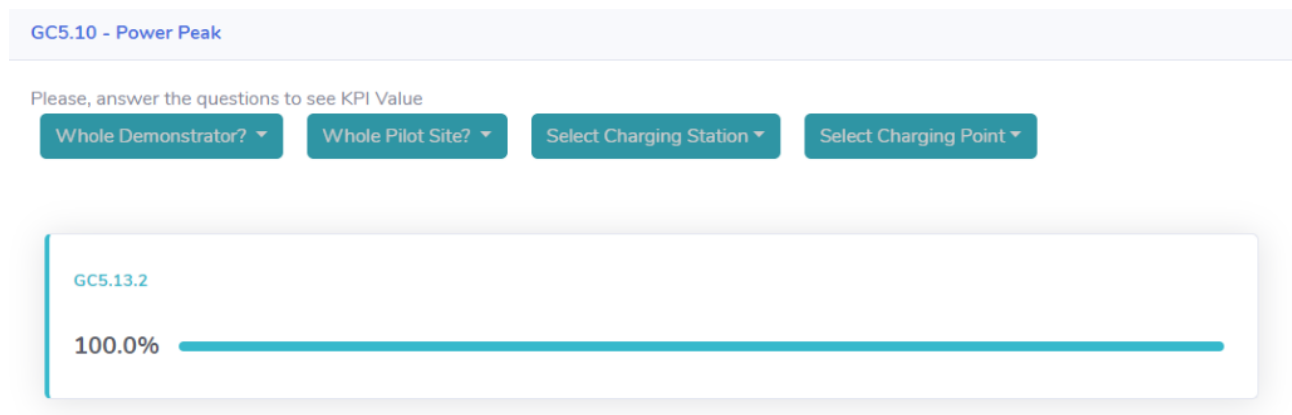


Figure 41 Power Peak KPI

As for the business KPIs, for their visualization it is necessary to select the input information with the drop down menus in Figure 42 which correspond to location, demo and year. Three graphs will be displayed as in Figure 42 the first which concerns the KPI which are natural numbers, the second which concerns KPI measured in Euro and the third which instead shows the business KPIs expressed as a percentage. The KPI are shown per month and each color corresponds to a different KPI. This view allows you to see how the KPI changes over the months.

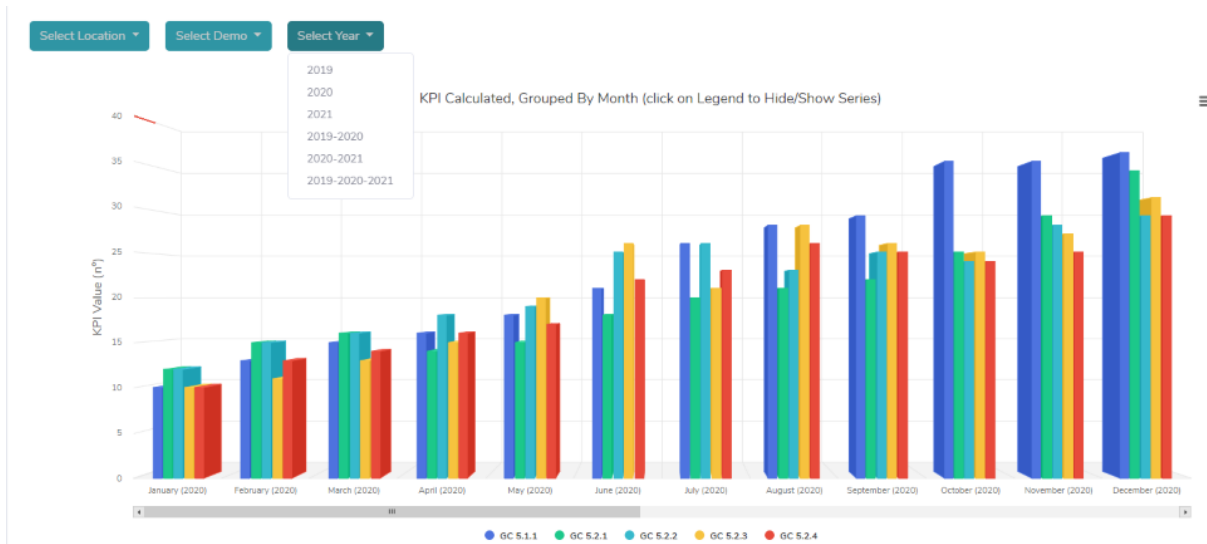


Figure 42 Business KPI example

5.5. Software Technology Used

The GreenCharge visualization tool is implemented in PHP and JavaScript based on Bootstrap for aspects related to the graphical design.

Bootstrap is a free front-end framework for faster and easier web development. Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins to easily create responsive designs.

For its operation, the web part of the application imports a considerable amount of JavaScript libraries like:

- HighCharts
- CanvasJS

Highcharts is a charting library written in pure JavaScript, free for non-commercial purpose, offering an easy way of adding interactive charts to web site or web application. Highcharts currently supports line, spline, area, areaspline, column, bar, pie, scatter, angular gauges, area range, areaspline range, column range, bubble, box plot, error bars, funnel, waterfall and polar chart types.

CanvasJS offers mostly similar features.

6. Deployment and utilization of the open source simulator

6.1. Design and use of the Container Based Deployment Configuration

The Container based deployment configuration allows for an easy deployment of the simulation platform on the user's workstation, independently from the Operating System. The software architecture is shown in Figure 43.

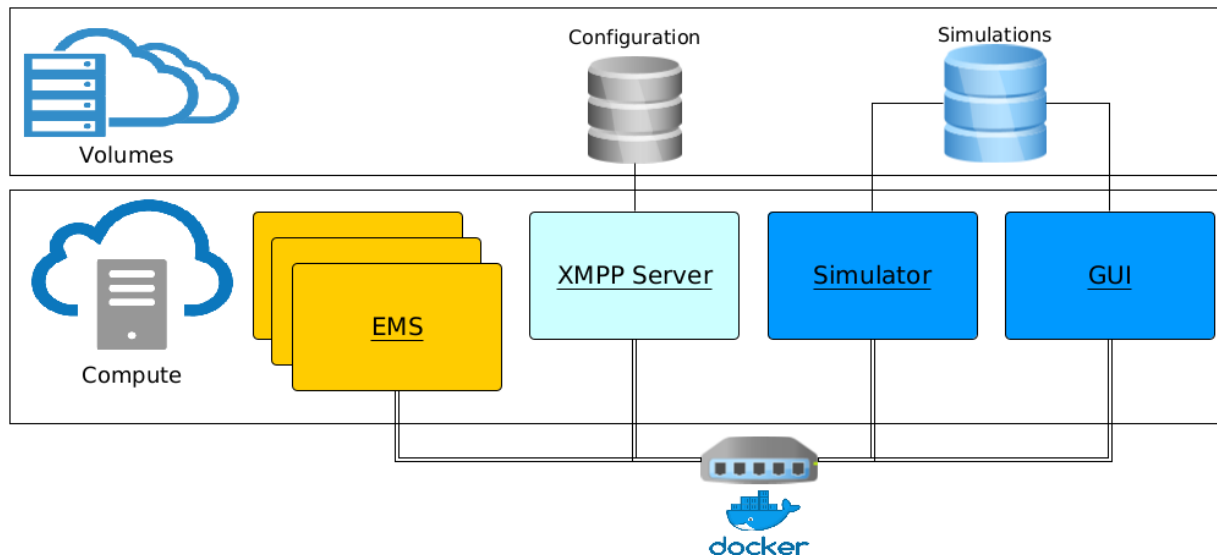


Figure 43 Container based deployment configuration.

Using both a virtual or a real network many containerized components interoperate through a loosely coupled integration. The blue boxes represent the simulation engine and its Graphical User Interface (GUI). The two components use a volume to access simulation input and output data such as configuration of scenarios, time-series and results.

The XMPP server provides a peer-to-peer communication overlay for multi-agents distributed implementation. A volume is used to save user-credentials, as the simulator can be used by multiple users who can run their simulations in parallel, in one or in multiple containers.

An optimization model can be integrated as Energy Management Systems (EMS) that runs in its own container and uses the Simulator interface to receive simulation events and to return the optimal energy schedule. In particular, the GreenCharge project will evaluate two different EMS innovative technological solutions, developed by the University of Oslo and by the Eurecat partner. Here we investigate an alternative solution that is used also to demonstrate how the simulation platform works.

6.1.1. The open source GitHub repository

The simulator software is available through the GitHub link <https://github.com/greencharge/gcsimulator>.

The repository contains scripts to build local Docker containers, holding the different components shown in the previous sections. In particular, in addition to the source code of the simulator engine and of the GUI, a dummy EMS implemented in JavaScript is provided for testing purposes. It is the same software that runs in the web page of the integration tool. Several Docker containers concur, with their scripts, to instantiate the simulator, with or without the GUI.

The repository also contains the installation and user manual, the API documentation and the actual software. We invite the reader to refer to the GitHub repository to access the updated version of the software documentation.

6.1.2. How to build the simulation platform

The only requirements to build the simulation platform is the installation of docker and of docker-compose, on any Windows, Linux or Apple machine.

The easiest way to build the simulation platform is to use the docker-compose command.

docker-compose up

A docker-compose configuration is provided to download the images and to instantiate all the containers, which communicate on a virtual network.

Nevertheless, the user can build the container image locally. This is the recommended solution to work with the last updated image. In this case, the images must be built before running the docker-compose command:

```
cd Dockers/gcsim
docker build . --tag gcsim
cd ../Dockers/gcscheduler
docker build . --tag gcscheduler
docker-compose up
```

N.B.: At the state of art the GUI is provided only as a docker image in docker-hub

6.2. Running the Simulator

In order to run the simulator a command line client allows for setting the simulation scenario, starting, stopping and checking the status of the running simulation. A demo user and a simple simulation scenario are configured for testing purpose.

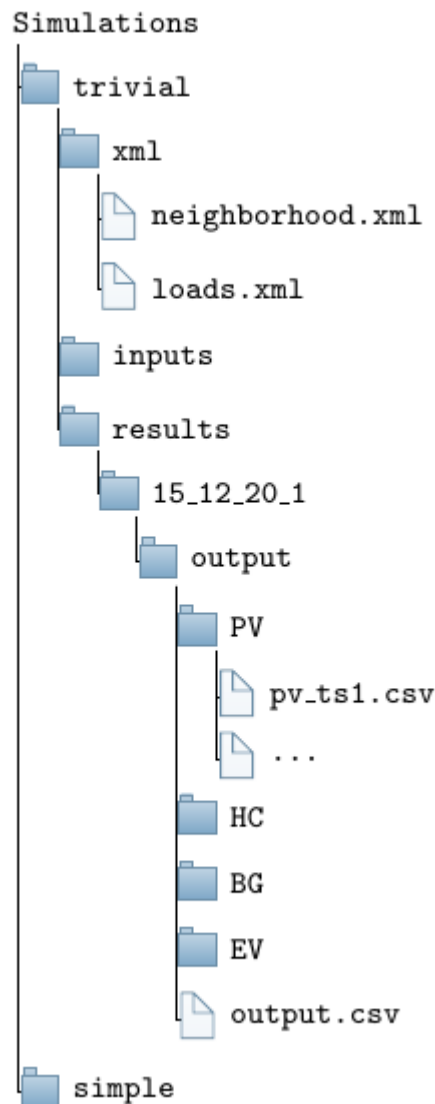


Figure 44 Folder tree of a simulation scenario.

In Figure 44, the folder tree of the available simulations is shown. To set the configuration to be simulated it needs to invoke the following command specifying the name of the simulation folder. In the current example, two scenarios are available: the trivial one and the simple one.

The execution of the simulator add in the result-folder a new sub-folder with output data. The name of the subfolder contains the date simulated day followed by an integer that is the simulation count.

6.2.1. For the Developer

For the developer the command invocation of the interface is provided to run the simulator. It allows for a headless execution that can be started inside or outside the container. Such a low-level interface will be exploited to automate the evaluation and to run the simulator in a remote container. Moreover, running commands inside the container does not need any additional requirement for the installation, just docker and docker-compose.

To run the simulator, it needs to execute the following commands, which start the dummy EMS and the simulation engine.

```
docker exec -it docker_gcscheduler_1 bash ./scheduler start  
docker exec -it docker_gcsimulator_1 bash ./starter.sh start
```

The status of the simulation can be checked using the following command

```
docker exec -it docker_gcscheduler_1 bash ./scheduler status  
docker exec -it docker_gcsimulator_1 bash ./starter.sh status
```

If the status of the dummy EMS is “stopped”, it means that the simulation has been completed and it needs to stop the simulation engine.

```
docker exec -it docker_gcsimulator_1 bash ./starter.sh stop
```

For the developer, a friendly command line python client that integrates the interface to select other EMS has been developed. Of course, in this case the user must install on his machine the python interpreter, at least v.3 version. We show here the client help, but we invite to refer to the developer guide at <https://github.com/greencharge/gcsim.git> for further and updated details.

```
$ gcclient --help  
usage: gcclient.py [-h] [--optimizer {eurecat,oslo,dummy}]  
                [--policy {green,cheapest,earliest}]  
CMD  
simulator client  
positional arguments:  
CMD                the daemon command  
optional arguments:  
-h, --help          show this help message and exit  
--optimizer {eurecat,oslo,dummy}  
                        start the scheduler  
--policy {green,cheapest,earliest}  
                        set the optimization policy
```

6.2.2. For the Evaluator

For the evaluator, a Graphical User interface is provided. Its extension with additional plugins for the automation of the evaluation is an ongoing work.

At the state of art, we provide a deployment configuration that allows for execution the GUI in a docker container, which can be accessed by a novnc web interface.

The strength of this deployment configuration is that the user does not need to install anything a part docker and docker-compose on its machine and can run the simulation using the browser.

The drawback is that it needs to download the image of a full desktop environment.

The GUI Container is based on “<https://hub.docker.com/r/dorowu/ubuntu-desktop-lxde-vnc/>” public image available on docker hub repository. This image is particularly useful since the configurator GUI was developed in python and would only be available via the operating system interface and not via the web. However, this

container allows the screen content of an O.S. to be displayed on a web page. This expedient allows the use of this GUI in a more flexible way and directly from a web browser. The docker containing the GUI is automatically launched at the execution of the docker-compose up command and exposes itself on a web page reachable by the browser at the address localhost: 6080. The web page will show an operating system where the simulator GUI is pre-installed and executable. Interface examples are shown later in the paragraph on how to create a scenario via the GUI.

An example of the web screen that will open when the container is run is the following:

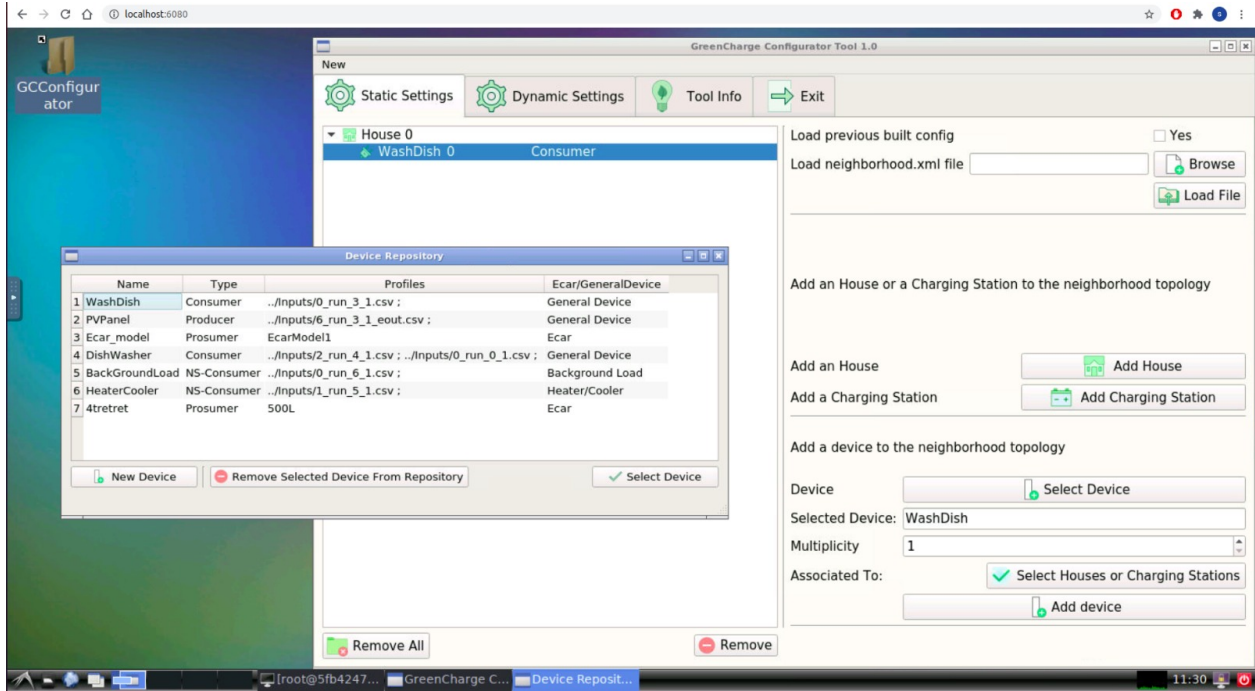


Figure 45 Web access to the GUI container.

As you can see from the screenshot in Figure 45, you have complete control of an operating system directly from a web browser.

The procedure to be carried out in order to create a new simulation scenario is described in the following paragraphs, in this paragraph instead we will show what happens when a new simulation session is started.

As you can see from the following Figure 46, through the "Control Panel" tab it is possible to start the last simulation created (the re-execution of old simulation scenarios is in development).

When the user presses on start, a start signal is sent to the simulator and scheduler which start their work.

In the GUI screen there is also a loading bar that shows the percentage of completion of the current simulation and given the stochastic nature of the schedulers, it is possible to select how many times you want to repeat the same simulation for a single configuration. There is also a pause simulation button which allows the user to pause a simulation. On the control page it is also possible to select the date to be simulated.

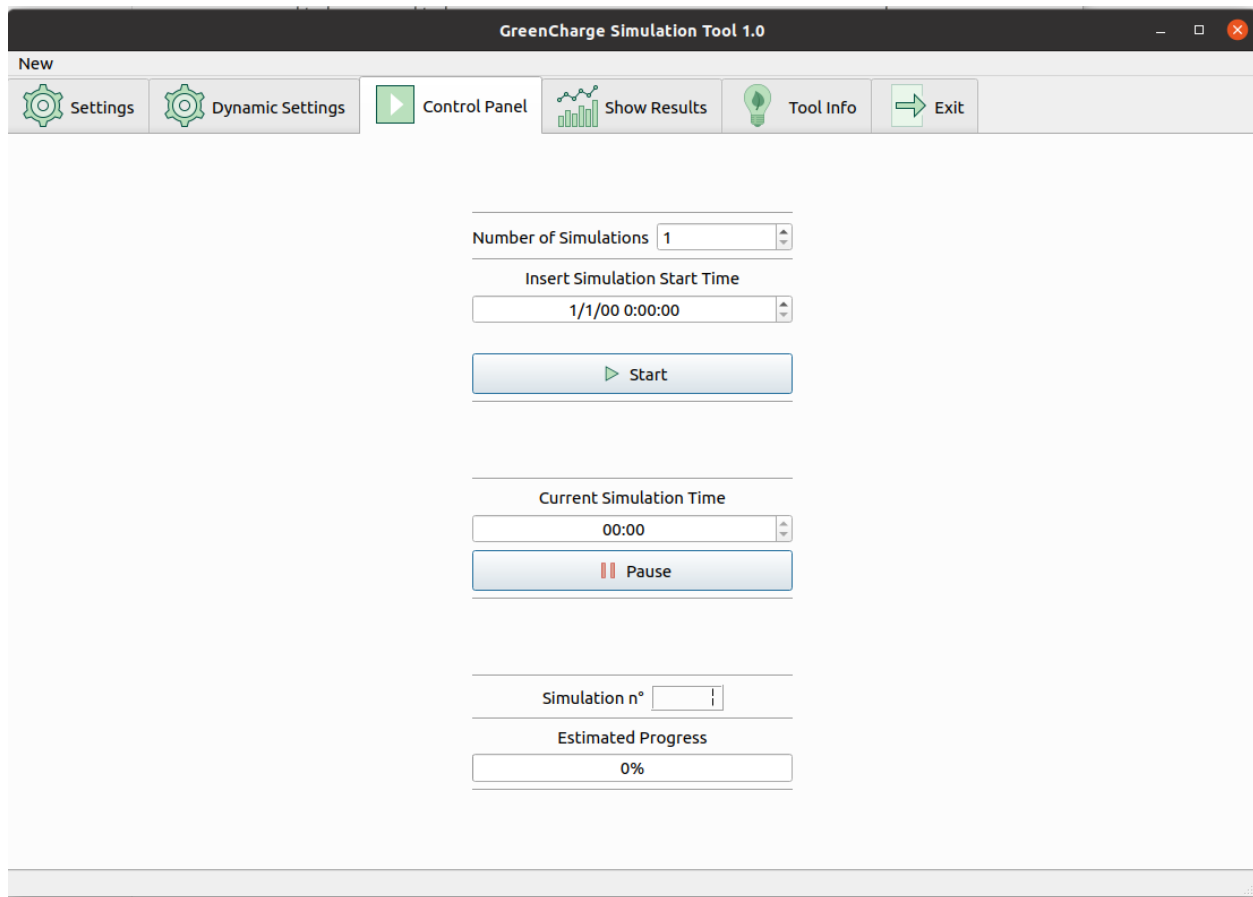


Figure 46 Control Tab

6.3. Visualization of Results

6.3.1. Report Generation

In order to generate a web report that summarizes the simulation results a specific command line utility is currently available. The report is exported as a web page that can be visualized by any browser offline. It can be distributed and shared as a standalone archive.

The report contains 5 sections. The **Parameters** section shows a list of synthetic parameters whose value is computed from the results of the simulation. For each parameter it eventually shows the expected value, if for that simulation scenario these values have been provided in advance for testing purpose. The list of parameters is shown in Figure 47.

Simulation Results			
parameters	Parameter	Simulated Value	Expected Value
checks	Total_Energy_Consumption	53463.6997646111	36547.03309794444
charts	Total_Energy_Production	17212.460872	17212.460872
kpis	Assigned Start Time List	[[{"t": 1449913400, "v": 1449900400}]]	[[{"t": 1449913400, "v": 1449900400}]]
	Energy_charged	23500.0	6583.333333333333
	Number_of_Timeseries	7	7
	Self_Consumption	1.0	1.0

Figure 47 Synthetic parameters of the simulation output.

Parameters

- Total Energy Consumption: It corresponds to the total consumption within the neighbourhood measured in W.
- Total Energy Production: It corresponds to the total production within the neighbourhood measured in W.
- Assigned Start Time List: It is a list of assigned start times, one for each device. The list is of the Key-Value type: the key is the identification of the device and the key is the timestamp of the assigned start time.
- Energy Charged: It is the energy in W charged by all electric vehicles in the simulation scenario.
- Number of Timeseries: It is the number of time series present in the simulation directory at the end of the same. It is assumed to be equal to the sum of the number of energy consumption and production events (sessions).
- Self-Consumption: It is the self-consumption's value. Self-consumption is defined as the division of self-produced energy consumed divided by the totality of energy produced by autonomous ecological sources.

The **Checks** section visualizes the result of coherence checking process. The list of checks is shown in Figure 48.

Simulation Results		
parameters	Coherency Rule	Result
checks	AsLstConstraintRespected	{}
charts	PowerPeaksReached	{'1': array([0.83907203]), '11': array([0.83098592])}
kpis	PowerPeaksLimits	{'1': '3000', '11': '3000'}
	PowerPeaksLimitsReached	{'1': 'not reached', '11': 'not reached'}
	Energy_Charged_respect_to_capacity	{'2': 'Respected'}
	Energy_Charged_respect_to_Connection	{'2': 'Respected'}
	Energy_AutoConsumed_Respect_To_Energy_Produced	Respected
	Charging_Power_Lower_than_Maximum	{'2': 'Respected'}

Figure 48 Coherence checking results.

Coherence Checks

- Est-Lst Constraint Respected: It is a Key-Value list that contains the id of a schedulable device as a key and a boolean as a value. The boolean is true if the assigned start time is between the minimum start time and the maximum start time, false otherwise.
- Power Peaks Reached: It is a Key-Value list that contains the id of a device as a key and a double as a value. The double corresponds to the peak of power consumed in a device's consumption profile.
- Power Peaks Limits: It is a Key-Value list that contains the id of an energy hub as a key and a double as a value. The double corresponds to the upper limit of the peak power that can be reached by an energy hub.

- *Power Peaks Limits Reached:* It is a Key-Value list that contains the id of an energy hub as a key and a boolean as a value. The boolean turns out to be true if, given the sum of the time series of all consumers belonging to an energy hub, it has a peak power value greater than or equal to the peak power limit of the energy hub itself. The maximum of the function generated as an interpolation of the sum function of the time series is considered as the power peak.
- *Energy Charged Respect to Capacity:* It is a Key-Value list that contains the ID of an electric vehicle as a key and a boolean as a value. The boolean takes on a true value if the electric vehicle charges a total energy (measured in W) less than or equal to the entire capacity of its battery. It therefore returns false if the electric vehicle charged more than physically possible given the storage limits of its battery.
- *Energy Charged Respect to Connection:* It is a Key-Value list that contains the ID of an electric vehicle as a key and a boolean as a value. The boolean takes on a true value if the electric vehicle charges a total energy (measured in W) that is less than or equal to the maximum energy that can be charged by the vehicle itself in an interval of time. The maximum amount of energy that can be charged is equal to the maximum charging power of an electric vehicle multiplied by the time interval.
- *Energy AutoConsumed Respect to Energy Produced:* It is a Boolean value. It is true if the amount of self-consumed energy calculated is less than or equal to the totality of self-produced energy.
- *Charging Power Lower than Maximum:* It is a Key-Value list that contains the ID of an electric vehicle as a key and a boolean as a value. The boolean takes on a true value if the electric vehicle always charges at a power that is less than or equal to its maximum charging power.

The **Charts** section visualizes some charts which summarize the energy utilization by the neighborhood. In Figure 49 we see the power consumed from the grid, the self-consumption and the power consumed by scheduled devices.

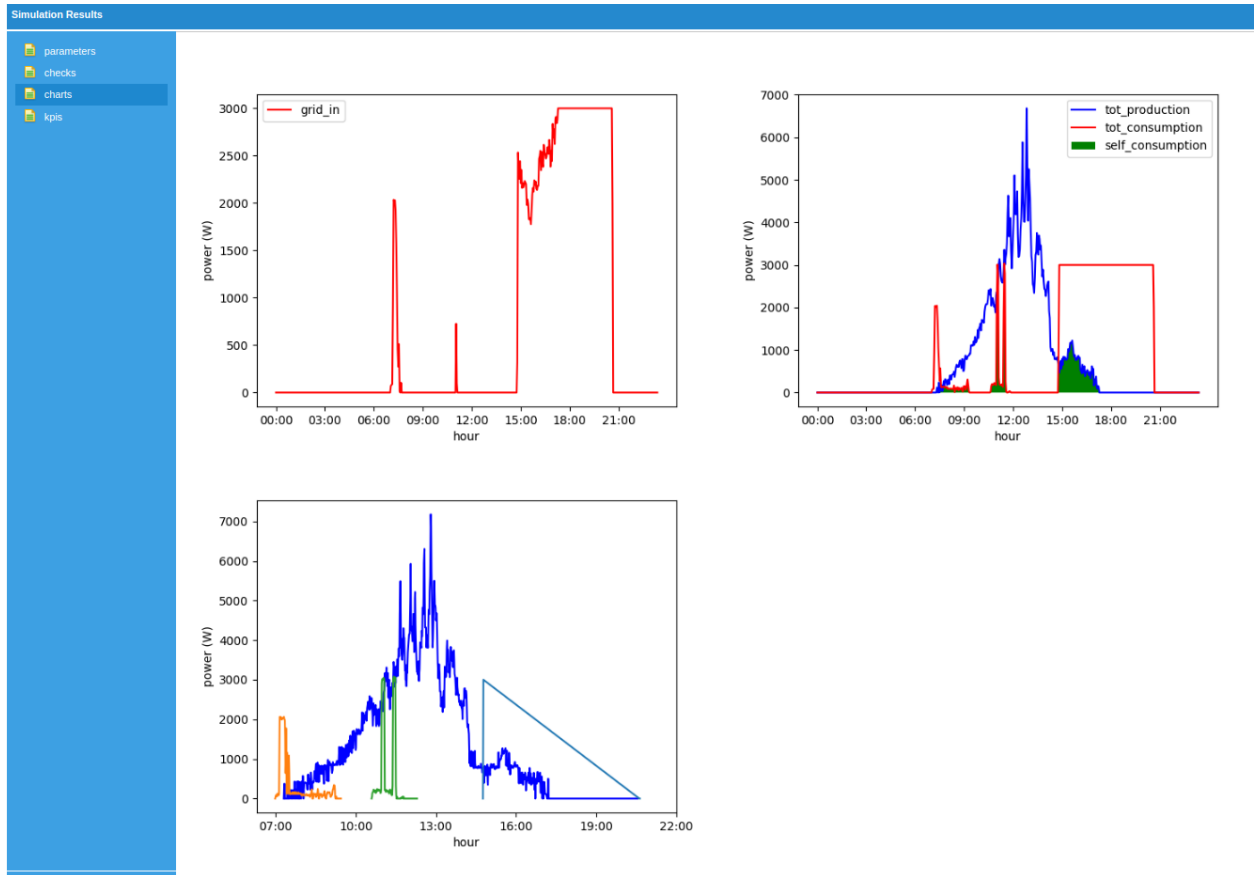


Figure 49 Visualization of times-series.

Finally, the **KPIs** list the energy related GreenCharge KPIs which can be evaluated from the output of simulation as shown in Figure 50.

Simulation Results			
parameters	ID	Name	Value
checks	GC5.1	Number Of EVs	1
charts	GC5.2	Number Of CP	1
kpis	GC5.14	Self Consumption	1
	GC5.3.1	Utilisation Of CPs	0.24305555555555555
	GC5.3.2	Utilisation Of CPs	1
	GC5.3.3	Utilisation Of CPs	100
	GC5.13.1	Charging Flexibility	0.9666666666666667
	GC5.13.2	Charging Flexibility	0.9999995172078835
	GC5.13.3	Charging Flexibility	0.9666666666666667
	GC5.4	Share Of Battery Capacity	31200000

Figure 50 Energy related KPI computation on simulation output.

6.4. Creation of a new Simulation Scenario

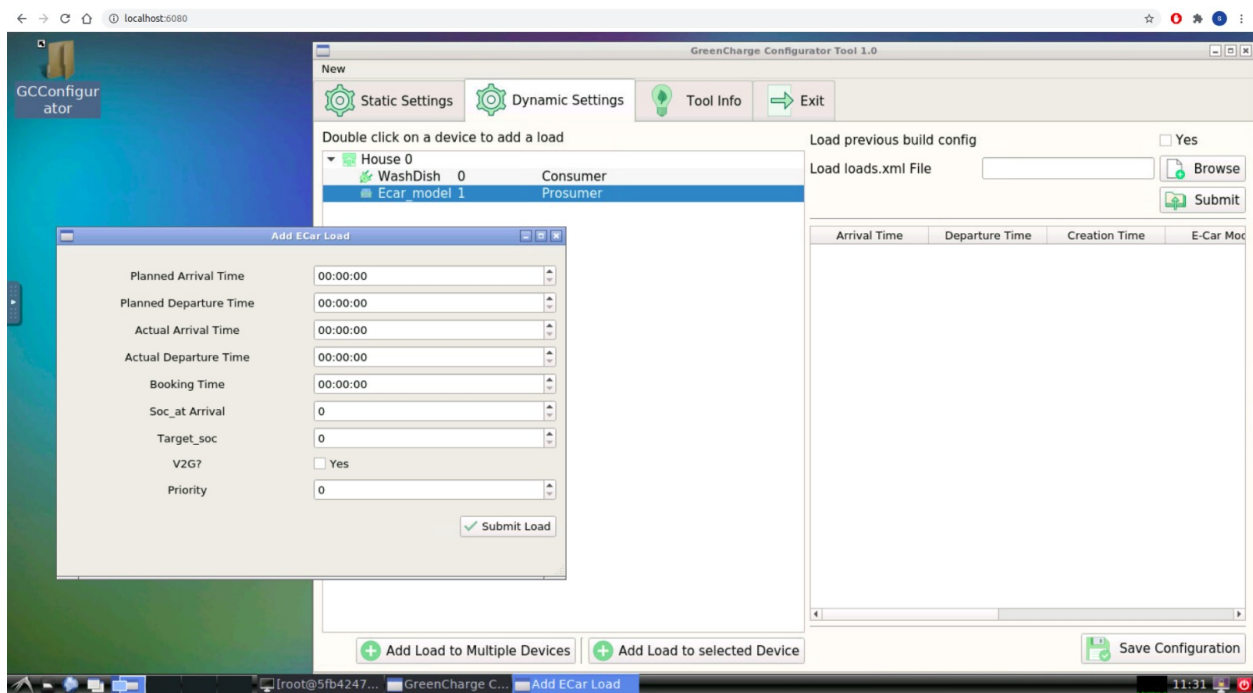
Configuration of a simulation scenario

As shown in the architecture figure, the simulator graphical interface docker is present in the interoperable docker system. The graphic interface allows the creation of scenarios and the setting of simulation parameters. To start the software, just go to the GCConfigurator folder on the desktop and run the command: `python3 starter.py`. A screen like the following will appear:

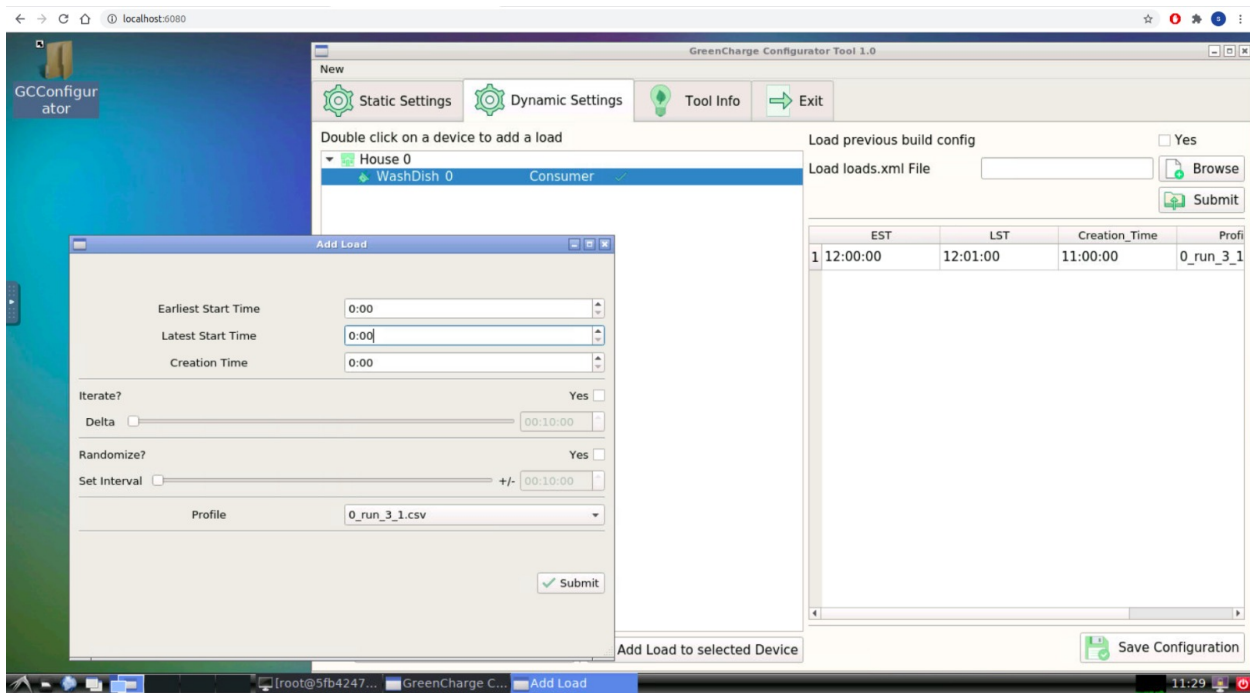
As you can see from the screen early provided in an early section, the configurator is divided into two main TABs:

- Static settings
- Dynamic settings

In order to create a new configuration, you need to move to the static settings tab and start modeling the scenario topologically by inserting both device and containers that can be houses or charging points and the actual devices (shiftable loads, background loads, heater/cooler loads, batteries, electric vehicles). In this section it will therefore be necessary to define all the parameters that belong to the device itself (the parameters of the device model) and not to the single simulation. Note that when a device is created it is saved in a database and it will be possible to reuse it later by changing the parameters of the single session. Once the static part of the scenario has been configured, you move to the dynamic settings tab where you need to define the parameters that instead depend on the single simulation (the arrival time of an electric vehicle, the departure time, the percentage of charge with which you arrive at the charging station and the desired one).



Logically, the static parameters, as well as the dynamic ones, depend on the type of device and it will be possible to set different parameters depending on the device we want to configure.



Once our configuration is finished, you can press the save configuration button. Another simulation folder will automatically be created inside the directory tree shown in the previous section that respects the structure expected by the simulator, the input profiles are copied into the new simulation and the xml files relating to the scenario just created are generated.

This sharing system is possible because the simulator and the graphical interface share the same docker volume.

6.5. Example of Test Scenario

In this section we will deal with a simple test scenario that has been designed to verify the correctness of the communication between simulator and optimisers.

The scenario includes all the devices provided by the GreenCharge simulator:

- A solar panel
- A schedulable load
- An electric vehicle
- A Heater/Cooler device
- A battery
- A Background load

Description in detail:

- **Solar Panel**

Name	id	Type	Profile	Energy_cost
PVPanel	1	Producer	10_run_5_1_pv.csv	0_run_0_1.csv

- **Schedulable Load**

Name	id	Type	Profile	Est	Lst	Creation_Time
Dishwasher	3	Consumer	10_run_1_1_dw.csv	10:43	10:43	05:10

- **Electric Vehicle**

Name	EV1
Model	Volkswagen e-Golf
Type	Prosumer
Id	2
Capacity	24
Max charging Power AC	5.0
Max Charging Power DC	5.0
Max Discharging Power AC	5.0
Max Discharging Power DC	5.0
Max Allowable Energy	1
Min Allowable Energy	1
SbCh	0.1
SbDis	0.1
Charging Efficiency	0.9
Discharging Efficiency	0.8
Start Time	01:23
End Time	16:00
Creation Time	01:08
Status of Charge	8%
Target Status of Charge	80%

- **Battery**

Name	Battery1
Type	Prosumer
Id	5
Capacity	80
Max charging Power AC	3.6
Max Charging Power DC	3.6
Max Discharging Power AC	3.6
Max Discharging Power DC	3.6
Max Allowable Energy	1
Min Allowable Energy	1
SbCh	0.8

SbDis	0.7
Charging Efficiency	0.9
Discharging Efficiency	0.9
Planned Arrival Time	14:53
Planned Departure Time	20:43
Actual Arrival Time	14:53
Actual Departure Time	20:43
Creation Time	03:46
Status of Charge	20%
Target Status of Charge	90%
Vehicle To grid	Unable
Priority	Unable

- **BackGround Load**

Name	id	Type	Profile
BackgroundLoad	6	NS-Consumer	grid_in.csv

- **Heating/Cooling Device**

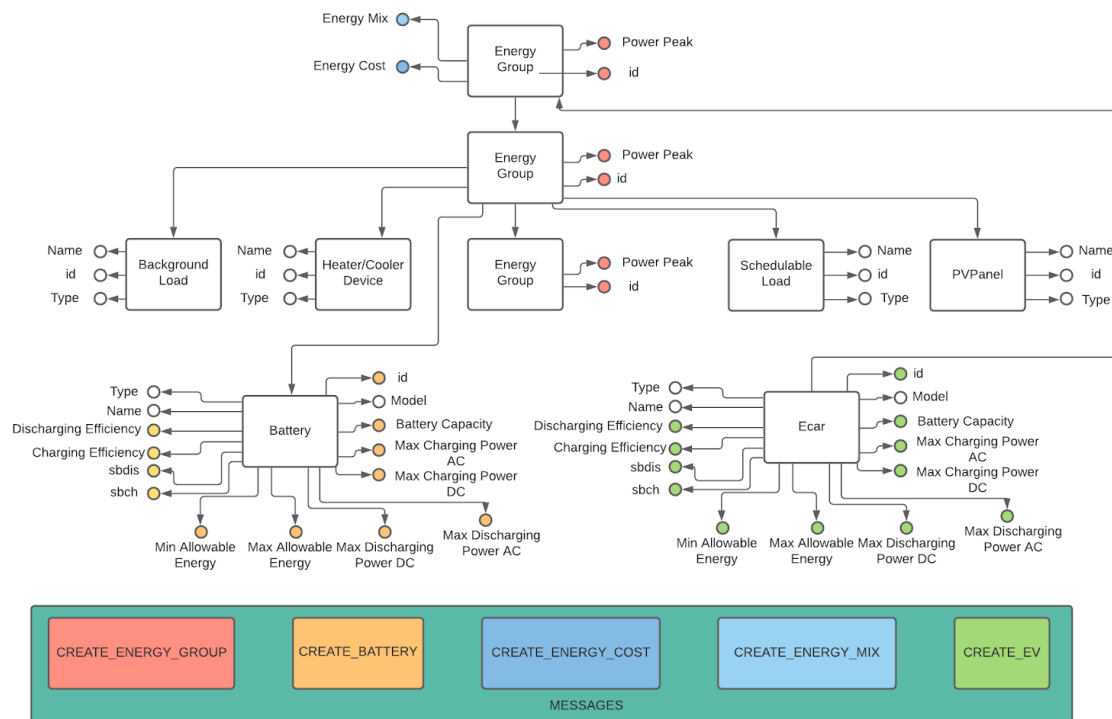
Name	id	Type	Profile
heatercooler1	7	NS-Consumer	7_freezerInput.csv

6.5.1. Configuration files

The following image shows the topological structure of the neighbourhood under examination and the attributes of the individual devices or entities present in the test scenario. The attributes of the devices are coloured in the same way as the message in which they are used.

In general, the messages that are generated with only the static parameters are configuration messages that do not reflect a real dynamic event but serve to inform the optimisers about the structure of the scenario. For example, the create EV message simply informs the optimiser that there is an electric vehicle in the scenario with the characteristics attached in the message but without the creation of the EV corresponding to a real event (arrival of the vehicle at the charging station, departure from charging station etc).

This section provides the xml file that encodes the scenario graphically presented in the image.



```
<?xml version='1.0' encoding='UTF-8'?>
<Neighborhood peakLoad="100">
  <energyCost>
    <id>5</id>
    <profile>cost.csv</profile>
  </energyCost>
  <energyMix>
    <profile>energy_mix.csv</profile>
  </energyMix>
  <house id="1" peakLoad="10">
    <user id="0">
      <device>
        <id>1</id>
        <name>PVPanel</name>
        <type>Producer</type>
      </device>
      <ChargingPoint id="0" peakLoad="60" ConnectorsType="DC">
```

```

<ecar>
<model>Volkswagen e-Golf</model>
<capacity>24</capacity>
<maxchpowac>3.6</maxchpowac>
<maxchpowcc>3.6</maxchpowcc>
<maxdispowac>3.6</maxdispowac>
<maxdispowcc>3.6</maxdispowcc>
<maxallen>1</maxallen>
<minallen>1</minallen>
<sbch>0.8</sbch>
<sbdiss>0.7</sbdiss>
<cheff>0.9</cheff>
<dis_eff>0.9</dis_eff>
<type>Prosumer</type>
<name>EV1</name>
<id>2</id>
  </ecar>
  </ChargingPoint>
  <device>

<type>Consumer</type>
<name>DishWasher</name>
<id>3</id>
  </device>

```

```

  <battery>
<model>battery1</model>
<capacity>80</capacity>
<maxchpowac>20</maxchpowac>
<maxchpowcc>30</maxchpowcc>
<maxdispow>40</maxdispow>
<maxallen>1</maxallen>
<minallen>0.1</minallen>
<sbch>0.1</sbch>
<sbdiss>0.1</sbdiss>
<cheff>0.9</cheff>

```



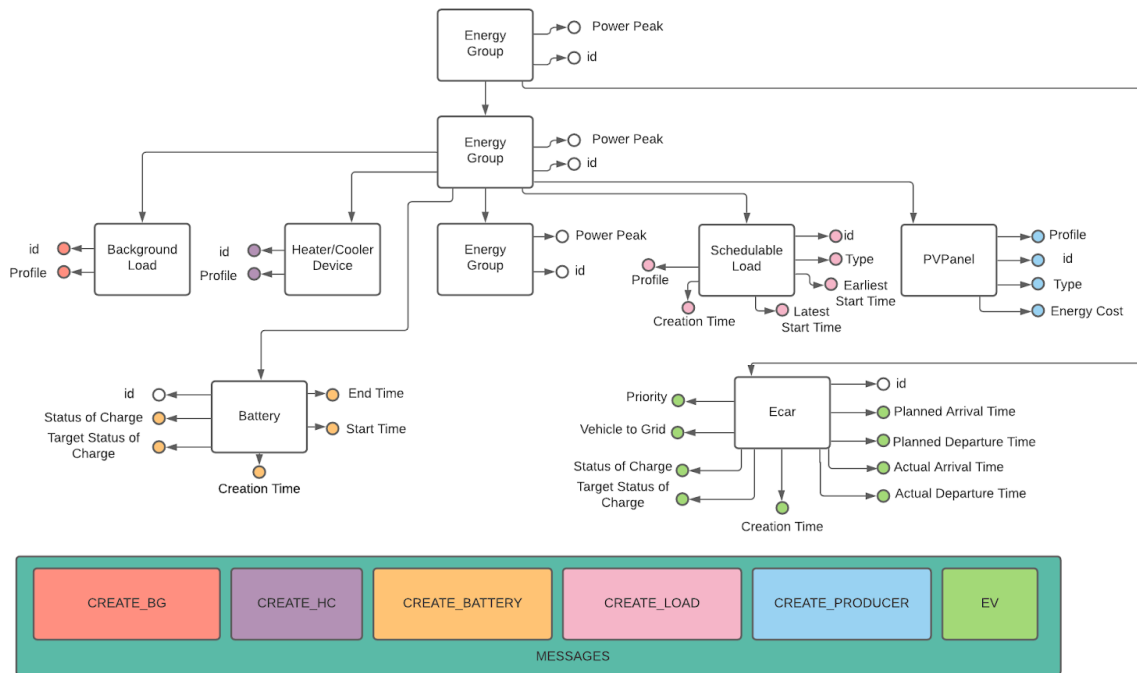
```

<dis_eff>0.8</dis_eff>
<type>Prosumer</type>
<name>batt</name>
<id>5</id>
  </battery>
  <backgroundload>
<type>NS-Consumer</type>
<name>BackGroundLoad</name>
<id>6</id>
  </backgroundload>
  <heatercooler>
<type>NS-Consumer</type>
<name>heatercooler1</name>
<id>7</id>
  </heatercooler>
  </user>
</house>
</Neighborhood>

```

Similarly to what was done for the static scenario part, the image is provided for the dynamic part of the simulation configuration (encoded in the load.xml file provided below).

The explanatory image shows the events with the same tree structure used previously. Here too the parameters of the devices are coloured with the same colour as the messages that contain them. Unlike the messages relating to the static structure, the messages that refer to the dynamic part correspond to real events that occur during the simulation; for example, a shiftable load that is entered into the system at 10 am or a charge request from an electric vehicle made at 11 am, the arrival of an electric vehicle at the charging station or the presence of an un-schedulable load entered the system at 3pm. All these events are converted into messages that the simulator dispatcher sends to the scheduler.



```
<?xml version='1.0' encoding='UTF-8'?>
<Neighborhood peakLoad2="10000">
  <house id="1" peakLoad="3000">
    <user id="0">
      <device>
        <id>1</id>
        <est>0</est>
        <lst>0</lst>
        <type>load</type>
        <creation_time>0</creation_time>
        <energy_cost>0_run_0_1.csv</energy_cost>
        <profile>10_run_5_1_pv.csv </profile>
      </device>
      <ChargingPoint id="0" peakLoad="3000" ConnectorsType="DC">
        <ecar>
          <id>2</id>
          <pat>50000</pat>
          <pdt>71000</pdt>
          <aat>50000</aat>
        </ecar>
      </ChargingPoint>
    </user>
  </house>
</Neighborhood>
</>
```

```

<adt>71000</adt>
<creation_time>10000</creation_time>
<soc>20</soc>
<targetSoc>90</targetSoc>
<V2G>0</V2G>
<priority>0</priority>
  </ecar>
</ChargingPoint>
<device>
<id>3</id>
<est>35000</est>
<lst>35000</lst>
<type>load</type>
<creation_time>15000</creation_time>
<profile>10_run_1_1_dw.csv </profile>
  </device>
  <battery>
<targetSoc>80</targetSoc>
<soc>8</soc>
<creation_time>500</creation_time>
<endTime>54000</endTime>
<startTime>1400</startTime>
<id>5</id>
  </battery>
  <backgroundload>
<id>6</id>
<profile>grid_in.csv</profile>
  </backgroundload>
  <heatercooler>
<id>7</id>
<profile>7_freezerInput.csv</profile>
  </heatercooler>
  </user>
</house>
</Neighborhood>

```

7. Conclusions

Based on the requirements analysis and design activities described in deliverable D5.2, the development of the prototype simulation and visualisation tools has advanced, allowing for the calculation of several KPIs, supporting the simulation and customization of complex scenarios and providing integration with different optimisers developed by partners in the GreenCharge consortium.

In particular, the following functional extensions/integrations have been provided:

- analytical models for energy demanding, producing and storing devices, such as batteries, pv panels and heaters/coolers;
- definition and implementation of interaction protocols for the integration of them simulator with optimisers, providing different optimization algorithms;
- clear and intuitive GUI for users, allowing them to seamlessly interact with the KPI calculator and simulation environment;
- precise and updated description of the software architecture and components.

The work carried out to finalize the design and implementation of the Simulation and KPI Calculation tools has demonstrated the importance of a strenuous collaboration among partners in the Consortium. Indeed, to meet the necessary requirements for future Evaluation steps, several interactions have been made to define a list of KPIs to be implemented in the calculator tool, and to be used as a reference for simulation. We expect that D5.4, focused on such Evaluation activities, will exploit the tools and gain benefits from them, especially in cases where actual data are not available or are insufficient. The integration of the developed tools with different external sources (i.e., data collected in the Pilots) will strongly support the evaluation phase of the innovation strategies experimented in the project, thus the simulator provides a clear interface and flexible scenario customization interfaces.

Another relevant aspect that has emerged regards the adopted technologies, which have demonstrated to be both a critical aspect of the integration step, and an important key to the resolution of many interoperability issues. Indeed, by applying standard technologies, which have been customized for the tools' needs, communications among interacting components such as optimisers and simulator have been made possible.

From the interaction with partners that will be involved in the Evaluation phase, the need has arisen for a quick-reference user guide, for both the Simulator and KPI Calculator tools, despite their interfaces being clear and intuitive. The guidelines, which cover both installation and usage of the tools, have been thus published online, and links have been provided in this deliverable. In this way, they could be updated in the future, if any changes need to be done to the tools or if a specific aspect of their interfaces needs to be further clarified.









8. External Links to Tools and Guidelines

Links to the external repositories in which the Tools described in this Deliverable have been reporting in this Section, together with links to Guidelines which can be used to operate them.

Repositories and links will be maintained for at least three years, ensuring they will always be reachable.

- KPI Calculator and Visualization Tools Guidelines:
<http://parsec2.unicampania.it/~branco/gccalculator/calculatorUserGuide.pdf>
- KPI Calculator and Visualization Tools repository:
<http://parsec2.unicampania.it/~branco/gccalculator/>
- GreenCharge Simulator guidelines:
<http://parsec2.unicampania.it/~branco/gccalculator/simulatorUserGuide.pdf>
- GreenCharge Simulator repository:
<https://github.com/GreenCharge/gcsimulator>

Members of the GreenCharge consortium

	<p>SINTEF AS (SINTEF) NO-7465 Trondheim Norway www.sintef.com</p>	<p>Project Coordinator: Jacqueline Floch, Jacqueline.Floch@sintef.no Technical Manager: Shanshan Jiang Shanshan.Jiang@sintef.no</p>
	<p>eSmart Systems AS (ESMART) NO-1783 Halden Norway www.esmartsystems.com</p>	<p>Contact: Terje Lundby terje.lundby@esmartsystems.com</p>
	<p>Hubject GmbH (HUBJ) DE-10829 Berlin Germany www.hubject.com</p>	<p>Contact: Jürgen Werneke juergen.werneke@hubject.com</p>
	<p>Fundacio Eurecat (EUT) ES-08290 Barcelona Spain www.eurecat.org</p>	<p>Contact: Regina Enrich regina.enrich@eurecat.org</p>
	<p>Atlantis IT S.L.U. (ATLAN) ES-08013 Barcelona Spain http://www.atlantisit.eu/</p>	<p>Contact: Ricard Soler rsoler@atlantis-technology.com</p>
	<p>Millor Energy Solutions SL (ENCH) ES-08223 Terrassa Spain www.millorbattery.com</p>	<p>Contact: Gerard Barris gbarris@enchufing.com</p>
	<p>Motit World SL (MOTIT) ES-28037 Madrid Spain www.motitworld.com</p>	<p>Contact: Valentin Porta valentin.porta@goinggreen.es</p>
	<p>Freie Hansestadt Bremen (BREMEN) DE-28195 Bremen Germany</p>	<p>Contact: Michael Glotz-Richter michael.glotz-richter@umwelt.bremen.de</p>



ZET GmbH (MOVA)
DE-28209 Bremen
Germany
www.zet.technology

Contact: Dennis Look
dennis@zet.technology



Personal Mobility Center Northwest
eG (PMC)
DE-28359 Bremen
Germany
www.pmc-nordwest.de

Contact: Bernd Günther
b.guenther@pmc-nordwest.de



Oslo kommune (OSLO)
NO-0037 Oslo
Norway
www.oslo.kommune.no

Contact: Paal Mork
paal.mork@bym.oslo.kommune.no



Fortum OYJ (FORTUM)
FI-02150 Espoo
Finland
www.fortum.com

Contact: Jan Ihle
jan.haugen@fortum.com



PNO Consultants BV (PNO)
NL.2289 DC Rijswijk
Netherlands
www.pnoconsultants.com

Contact: Simone Zwijnenberg
simone.zwijnenberg@egen.green



Università Degli Studi Della
Campania Luigi Vanvitelli (SUN)
IT-81100 Caserta
Italy
www.unicampania.it

Contact: Salvatore Venticinque
salvatore.venticinque@unicampania.it



University of Oslo (UiO)
NO-0313 Oslo
Norway
www.uio.no

Contact: Geir Horn
geir.horn@mn.uio.no



ICLEI European Secretariat GmbH
(ICLEI)
DE-79098 Freiburg
Germany
www.iclei-europe.org

Contact: Stefan Kuhn
stefan.kuhn@iclei.org
Innovation Manager:
Reggie Tricker
reggie.tricker@iclei.org