Internship Report

Lawrence Berkeley National Laboratory

Jonathan Coignard

September 1, 2015

I would like thank Dr. Samveg Saxena for the freedom he gave me in my work. I would also like to thank my charming and beautiful girlfriend for her love and support during my time in California. Enfin, je voudrais remercier mes parents Charles et Marie-Annick sans quoi rien n'aurait été possible.

Contents

List of Figures

List of Tables

1

Introduction

This document is intended to cover my work at the Lawrence Berkeley National Laboratory during my five and a half-month internship (February 13th to July 31st). It will also include the conclusions on this work experience within the Grid Integration Group (GIG).

1.1 Lawrence Berkeley National Laboratory

The Lawrence Berkeley National Laboratory has one mission statement: "Bring science solutions to the world"[1].

Berkeley Lab was founded in 1931 by Ernest Orlando Lawrence, a UC Berkeley physicist who won the 1939 Nobel Prize in physics for his invention of the cyclotron, a circular particle accelerator that opened the door to high-energy physics.

Berkeley Lab is a member of the national laboratory system supported by the U.S. Department of Energy through (DOE) its Office of Science. It is managed by the University of California (UC) and is charged with conducting unclassified research across a wide range of scientific disciplines.Located on an 82-hectare site in the hills above the UC Berkeley campus that offers spectacular views of the San Francisco Bay, Berkeley Lab employs approximately 3,232 scientists, engineers and support staff. The Lab's operating costs for the financial year 2014 total $750.9 million.

Berkeley Lab is a prestigious institution with 13 Nobel Prizes.

1.2 The Grid Integration Group (GIG)

The Grid Integration Group conducts research that advances the near-term adoption of demand response (DR) technologies, policies, programs, strategies and practices `https://gig.lbl.gov/`.

[1] Berkeley promotion video https://www.youtube.com/watch?v=G5nK3B5uuY8

The Grid Integration Group is part of the Energy Storage and Distributed Resource area (ESDR), which is itself inside ETA (Energy Technology Area)[2]. The hierarchy diagram (Figure 1.1) represents the ETA structure at the higher level, where the second hierarchy diagram (Figure 1.2) represents the structure of the ESDR.

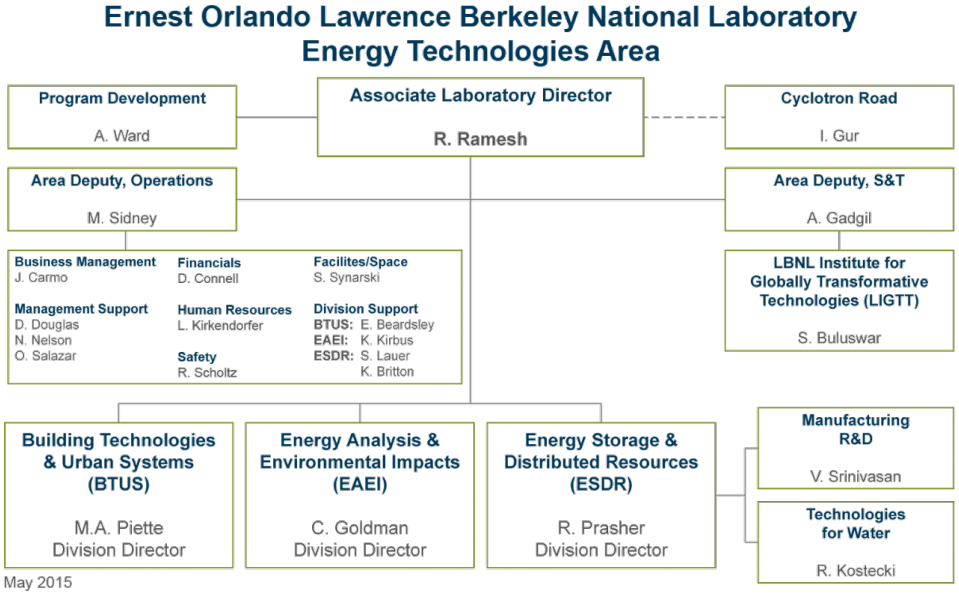


Figure 1.1: ETA Hierarchical structure

The GIG host Dr. Samveg Saxena is conducting research on the impact of EVs on the grid. I have been working as part of Dr. Saxena's team that during my internship. My focus has been on the development of MyGreenCar platform and V2G-Sim research software.

1.3 Vehicle-to-Grid (V2G) concept

With the current challenges related to the global warming and economic growth of the cities, producing clean energy is becoming a major concern. Electrical vehicles (EVs) and renewable energies might be part of the solution to achieve more sustainability. This report will focus on systems to enable clean transportation (and eventually a clean grid).

The pollution coming from vehicles in the United States represents more than 50% of the nitrogen oxide (NOx) and 75% of carbon monoxide emissions according to the Environmental Protection Agency (EPA)[3]. In this

[2] http://eetd.lbl.gov/about-us/organization/environmental-energy-technologies-division

[3] `http://www.epa.gov/airquality/peg_caa/carstrucks.html`

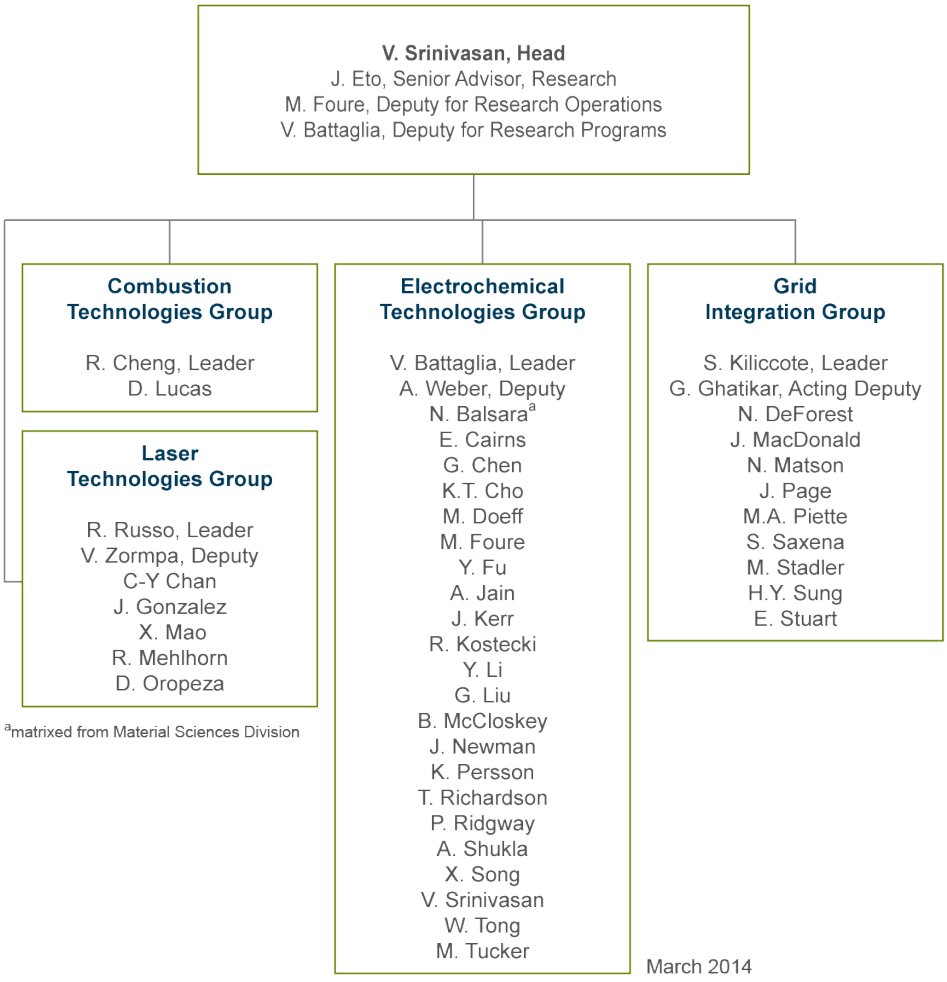


Figure 1.2: ETA researchers in the Energy Storage and Distributed Resources area (ESDR)

context, EVs could help reduce emissions as they do not directly produce harmful components. However, EVs are just shifting the problem on the power plant producing the electrical energy.

Renewable energy production is progressively increasing everywhere facing new availability challenges. For instance, in California, the excess solar production could cause problems during the day as shown on Figure 1.3. The "Duck curve" as it is called is remarkable, because of its the valley created from the solar energy production. In this scenario, EVs could play a major role in absorbing the excess production to redistribute it in the evening. Electrical vehicles would then enable a cleaner grid, which is also key for a cleaner transportation.

The battery equipped in EVs are also in progress, including the price of

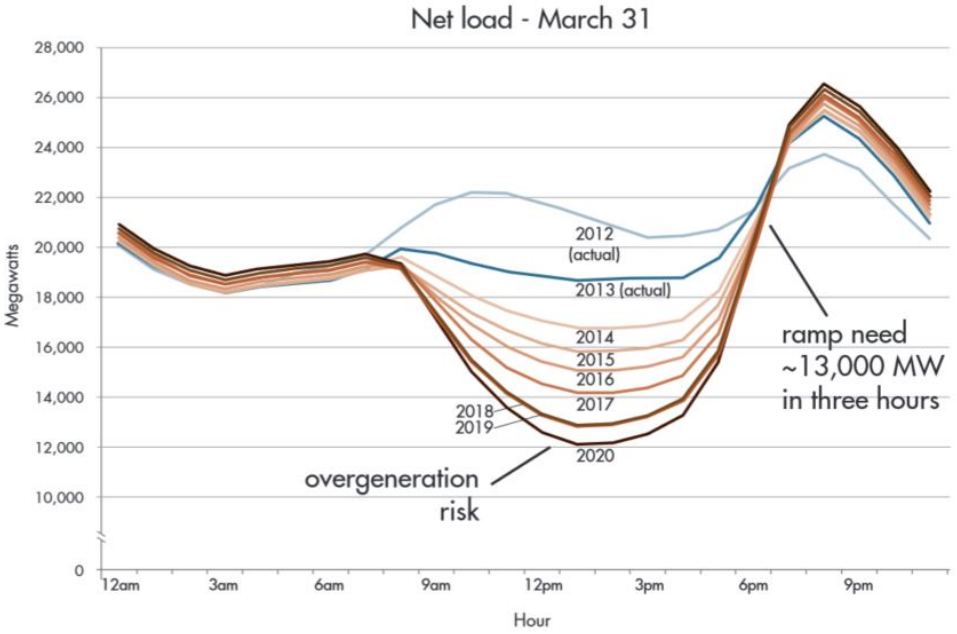


Figure 1.3: Californian "Duck Curve" representing the net load

a Li-ion battery packs declined from above $1,000 to around $410 per kWh between 2007 and 2014[4].

The Vehicle-to-Grid (V2G) concept is defined by a system in which plug-in electric vehicles interact with the power grid by regulating loads and delivering electricity. As vehicles are parked 90% of the time, it seems a good opportunity to use vehicle energy to balance the grid, and eventually generate profit. MyGreenCar, and especially V2G-Sim intend to answer some of the V2G challenges:

- What kind of charging infrastructure is needed city wide ? How much power should be delivered ?
- What kind of control algorithm would ensure an optimal use of the EVs battery ?
- At what price grid services should be rewarded ?
- ...

[4] `http://journalistsresource.org/studies/environment/energy/electric-vehicles-battery-technology-renewable-energy-research-roundup`

Report Structure

The report will start with the MyGreenCar initiative to help the deployment of electrical vehicles (EVs). The third chapter will focus on the Vehicle to Grid Simulator (V2G-Sim) which intends to analyze the impact of millions of EVs on the grid. Finally, I will discuss my experience here at the Lawrence Berkeley National Laboratory.

2

MyGreenCar

2.1 MyGreenCar concept

Drivers and fleet managers are often unaware of the benefits of clean vehicles, and many are unaware whether these vehicles even meet their personal mobility needs. MyGreenCar seeks to increase the adoption of clean vehicles by enabling drivers and fleet managers to understand whether clean vehicles meet their personal needs, and quantify the magnitude of fuel and cost savings from choosing a clean vehicle. MyGreenCar provides personalized information on how clean vehicles benefit individual drivers based on their unique travel patterns, route-specific terrain and traffic conditions, and driving style. MyGreenCar collects driver mobility data and, via server-side high-fidelity vehicle powertrain models, provides rich personalized information to quantitatively show how a particular clean vehicle will benefit a driver.

The MyGreenCar system consists of a smartphone and/or web interface, and a server-side system of physics-based powertrain models. Following a 3-step process, MyGreenCar provides a decision-support tool for new car buyers to enable informed purchasing of clean vehicles:

1. Users record their travel patterns and personalized drive cycles for any duration.

2. Users choose vehicle types and data is transmitted to a V2G-Sim server that uses physics-based powertrain models to predict SOC, charging needs, fuel consumption, etc. Models consider important factors such as traffic, terrain, ancillary power consumption, etc.

3. Analysis results are reported to the user via an interactive website.

2.1.1 Applications of MyGreenCar

Enabling personalized fuel economy predictions

The US Environmental Protection Agency (EPA) currently uses a 5-cycle test procedure to determine the fuel economy label values, however there is criticism of this approach because the drive cycles of individual drivers can vary greatly.

MyGreenCar enables drivers to predict personalized fuel economy, specific to their own driving patterns, routes, terrain, etc., across many vehicle models and types. Drivers can predict the fuel consumption and operating costs for conventional vehicles, hybrids, plug-in hybrids, electric vehicles, or fuel cell vehicles.

- How do these cars compare in terms of fuel economy and operating costs for my personal driving patterns and style?
- Are there comparable cars available that will offer greater fuel economy for my driving style?

Eliminating EV Range Anxiety for Prospective EV Buyers

Range anxiety, the concern that a vehicle will not provide the charge necessary for a driver's trips, is a principal challenge for widespread adoption of EVs – this problem is solved by MyGreenCar.

- Does an EV provide the range requirements for my normal travel patterns? Am I ever in danger of running out of charge?
- How much range will I have if I need to make an unexpected trip?
- What kind of charging infrastructure do I need ?

Eliminating EV Range Anxiety for Current EV Owners

Range anxiety for current EV owners is manifest in the form of uncertainty in whether a vehicle has enough charge for a desired trip. MyGreenCar can eliminate range anxiety for current EV owners by quantitatively answering the following questions:

- At my car's current SOC, do I have enough charge to make my next trip? (when considering traffic, terrain, ancillary power consumption, etc.)
- If not, how much do I need to charge my car? How long will this charging take on different types of chargers?

Informing Clean Vehicle Purchasing for Fleet Vehicles

The decision to purchase cleaner, but more capital-expensive, fleet vehicles is difficult because fleet managers are lacking the information and tools to forecast the magnitude of fuel savings offered by clean vehicles, and the resulting return-on-investment time.

- How much fuel savings will an advanced vehicle get in comparison to a conventional vehicle on our routes?
- How long will it take to recoup the added capital cost of an advanced vehicle?
- Which routes should I deploy the advanced vehicle on to capture the greatest fuel savings benefits?

Driving Data Database for Research

The data collected by MyGreenCar can provide a valuable resource for researchers and analysts in transportation planning, vehicle design, and many other fields. The MyGreenCar database will grow substantially with each year of operation, and thus a database structure will be established to organize, maintain, and filter the historical data. All data will be appropriately anonymized to protect the privacy of MyGreenCar users.

Spatially Forecasting PEV Deployment for Grid Infrastructure Planning

As plug-in vehicle deployment grows, utilities will need to upgrade distribution systems infrastructure to accommodate the increased load from PEVs. Spatially-resolved foresight into the charging load from PEVs is valuable for utilities to plan their system upgrades. MyGreenCar can provide PEV charging load forecasts for utilities that are both temporally- and spatially-resolved.

Greater deployment of PEVs represents an opportunity to simultaneously enable clean transportation and a clean electricity grid. Using PEV batteries for grid storage requires predictive information on the charging loads, battery SOC levels, flexibility to provide grid services and the approximate location of each vehicle throughout the day. The MyGreenCar app provides the foundational data collection capability to tackle those challenges.

2.2 MyGeenCar technical realisation

Possibly the most exciting thing about smartphone technology is that the field is still wide open [1]. Smartphones are part of everybody daily life, they are also surprisingly stuffed with a long list of captors (accelerometer, gyroscope, magnetometer, proximity sensor, light sensor, barometer, thermometer, air humidity sensor, pedometer, heart rate monitor, fingerprint sensors, microphone and cameras)[2]. Smartphone enable researchers to collect meaning-full insight on people's behavior. This is the reason why MyGreenCar systems are based on smartphone data collection. Nevertheless smartphones raise some issues along the data collection:

- How to categorize useful data from none representative inputs?
- How to account for accuracy in the data collection?
- How to ensure privacy and security of the users?
- How to ensure a good experience when using MyGreenCar?
- How to get people's feed back?
- ...

This section intends to present from A to Z the creation of the MyGreenCar platform. As there is no such thing as the perfect tool, a special focus will be brought to the choices being made along the way.

The Figure 2.1 describes how the information travel from an end user to our server and back to the client. At the beginning the user can press "start" or let his app automatically start the trip. After a while and once the user reach his destination the trip is sent to our server through a specific URL. The MyGreenCar server recognize the user and serialize the raw data into our database. Once data is safely stored an analysis request is triggered and launch a vehicle power-train model. Our models are then able to compute the raw data and save the corresponding fuel (or battery) consumption back into the database. At the end of this process the results store in the database are flowing back to an user via an email or through our interactive website.

Before we start going into the Django project, it is import to mention that the server run under Ubuntu. For more insight about the software behind the scene consult `http://michal.karzynski.pl/blog/2013/06/09/django-nginx-gunicorn-virtualenv-supervisor/`.

[1] `http://electronics.howstuffworks.com/smartphone5.htm`

[2] `http://www.phonearena.com/news/Did-you-know-how-many-different-kinds-of-sensors-go-inside-a-id57885`

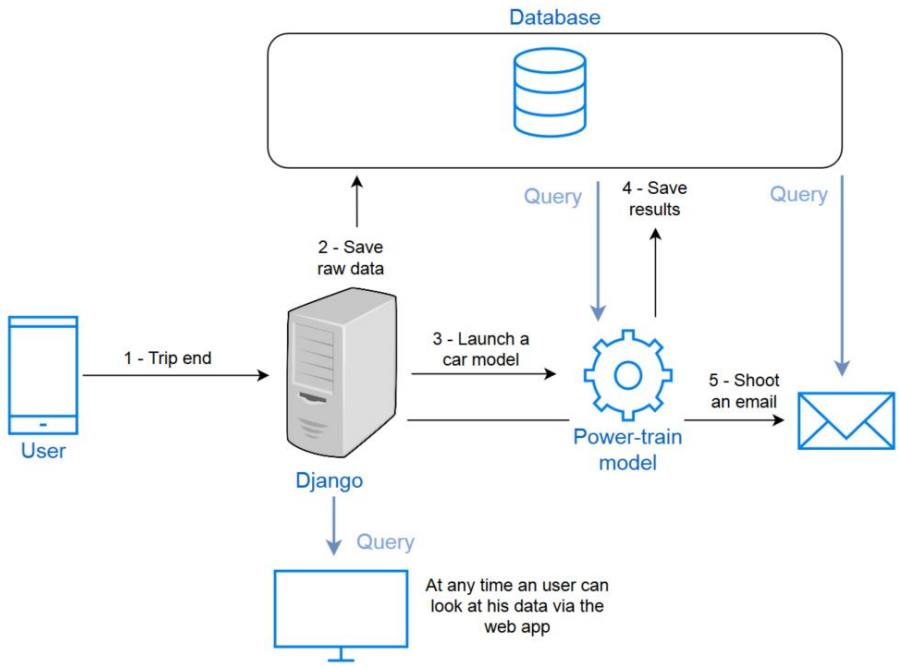


Figure 2.1: MyGreenCar system overview

2.2.1 Django at the heart of the server

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design (`https://www.djangoproject.com/`). At its best, Web development is an exciting, creative act; at its worst, it can be a repetitive, frustrating nuisance. Django lets you focus on the fun stuff – the crux of your Web application – while easing the pain of the repetitive bits[3]. Django is a widely use tool with a impressively growing community. For instance Instagram, Mozilla Support, Bickbucket, ... are based on Django [4]. But what is Django ?

Figure 2.2: Django web-framework logo

[3]source: `http://codecondo.com/popular-websites-django/`
[4]`http://codecondo.com/popular-websites-django/`

High level concepts

Django is designed to work in a slightly modified Model-View-Controller (MVC) pattern (Figure 2.3). MVC is a software engineering architecture concept describing how a program can change the visual aspect and the business aspect without affecting one another[5]. In Django's case, the "Controller" is probably the framework itself: the machinery that sends a request to the appropriate view, according to the Django URL configuration. The original Django developers refer to it as a model, template and view (MTV) to quote the Django documentation:

- "the view describes which data you see, not how you see it"
- "a model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you're storing. Generally, each model maps to a single database table"
- "Django's template engine provides a powerful mini-language for defining the user-facing layer of your application, encouraging a clean separation of application and presentation logic"

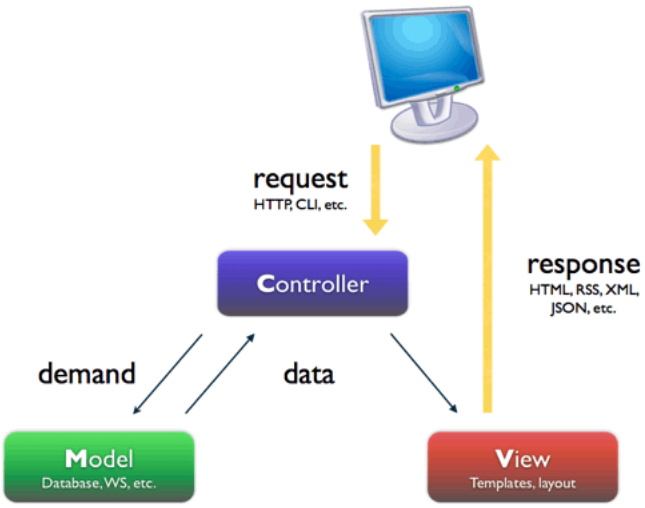


Figure 2.3: Model-View-Controller (MVC) pattern

MyGreenCar Models

The models are directly representative of our postgreSQL database. PostgreSQL is a open source object-relational database system. For the story, it has been created in University of California at Berkeley (UCB) by Michael

[5] http://www.webmonkey.com/2010/02/get_started_with_django/

Stonebraker as a followup project to its predecessor Ingres, "Post-gres" was borned [6].

The Figure 2.5 represents the structure of MyGreenCar database. This UML diagram is translated into a real PostreSQL database through the models.

```
class AnalysisRequest(models.Model):
    """
    Relates to one or several driving routes with a specific car
    """
    # An analysis can contains many routes
    driving_route = models.ManyToManyField(DrivingRoute,
                                related_name='analysisrequest',
                                null=True, blank=True)

    # One Analysis is done with a single car
    car = models.ForeignKey(Car, related_name='analysisrequest',
                            blank=True, null=True)

    def __unicode__(self):
        return u'%s %s' % (str(self.id), str(self.car))
```

Figure 2.4: Django implementation of the Analysis request table

The python code Figure 2.4 show the implementation of the "analysis request" model. Django will then transform the model into SQL language in order to create the tables within PostgreSQL. As we can see on Figure 2.5 the "request analysis" model has a many to many relationship with the "driving route" and a one to many relationship with a car.

[6] http://www.postgresql.org/about/history/

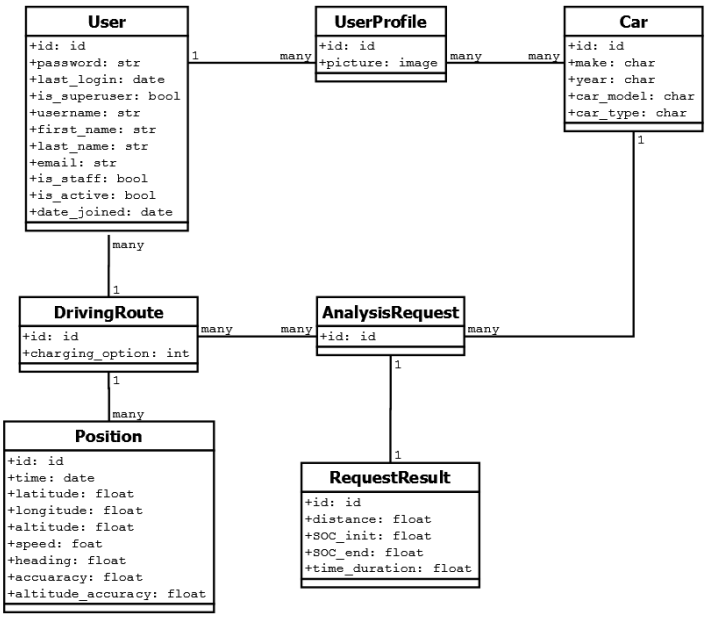


Figure 2.5: UML diagram of MyGreenCar database

MyGreenCar Views

As mentioned previously, the view describes which data you see, not how you see it. The scope of "views" is even wider as it is also in charge of receiving the data from end users. In the latter, the view is linked to a serializer in order to create the proxy between the JSON objects sent from smartphone and the database. Django can be completed with the REST framework[7], so that certain task as serializing and authentification are made easier. Once the data has been serialized the view will also triggered a method launching our power-train models.

With the REST framework instead of being simple function views become inherited class with their own methods. The following example shows the entry point of any data on the server (`http://mygreencar-01.lbl.gov/greencar_app/routes`). Serializing the data is mostly handle as a back-end by the REST framework, however the "post_save" function describes (through the comments) the process after an user submit a trip.

[7] `http://www.django-rest-framework.org/`

```
class DrivingRouteViewSet(viewsets.ModelViewSet):

    serializer_class = DrivingRouteSerializer
    permission_classes = (OwnerObjectViewPermission,
                          permissions.IsAuthenticated)

    def get_queryset(self):
        return DrivingRoute.objects.filter(
                            user=self.request.user)

    def pre_save(self, obj):
        obj.user = self.request.user

    def post_save(self, obj, created):
        # ID created by Django to represent this driving route
        id_drivingroute = obj.id

        # Default car (id=2) otherwise user favorite car
        car_instance = Car.objects.get(id=2)
        car_list =
            UserProfile.objects.get(user=obj.user).car.all()
        if car_list:
            car_instance = car_list[0]

        # Create an analysis request
        # associate to a car instance and a driving route
        auto_analysis = AnalysisRequest(car=car_instance)
        auto_analysis.save()
        auto_analysis.driving_route.add(id_drivingroute)
        auto_analysis.save()
        singletrip = '1'

        # Get the model and the car type
        model = car_instance.make.lower()
                + "_"
                + car_instance.car_model.lower()
        carType = car_instance.car_type

        # Launch one of our powertrain models
        mygreencar.delay(obj.user.username, obj.user.email,
    model, carType, singletrip, auto_analysis.id)
```

MyGreenCar Templates

At the time I writing those lines, MyGreenCar appearance is still under heavy development. Nevertheless I will‘ share the path we have decided to take.

First of all in order to easily deploy slick and nice website pages, we have decided to use Bootstrap `http://getbootstrap.com/` originally created by a designer and a developer at Twitter.

Secondly, to bring some dynamic into our webpage Angular JS is a good

option. AngularJS is a JavaScript MVC framework developed by Google that lets you build well structured, easily testable, and maintainable front-end applications. It extends HTML by providing directives that add functionality to your markup and allow you to create powerful dynamic templates[8]. Angular JS achievements are available at `https://builtwith.angularjs.org/`.

Finally, to display in a meaning full way user's data D3.js is a wonderful JavaScript library (`http://d3js.org/`). I would advise anybody who wants to represent data in a web browser to have a look at their examples. Coupling D3 representation with Google map API is a key feature envisioned by MyGreenCar development team. About creating map with D3 the following links give some ideas `http://www.toptal.com/javascript/a-map-to-perfection-using-d3-js-to-make-beautiful-web-maps` and `http://blog.cartodb.com/cartodb-makes-d3-maps-a-breeze/`.

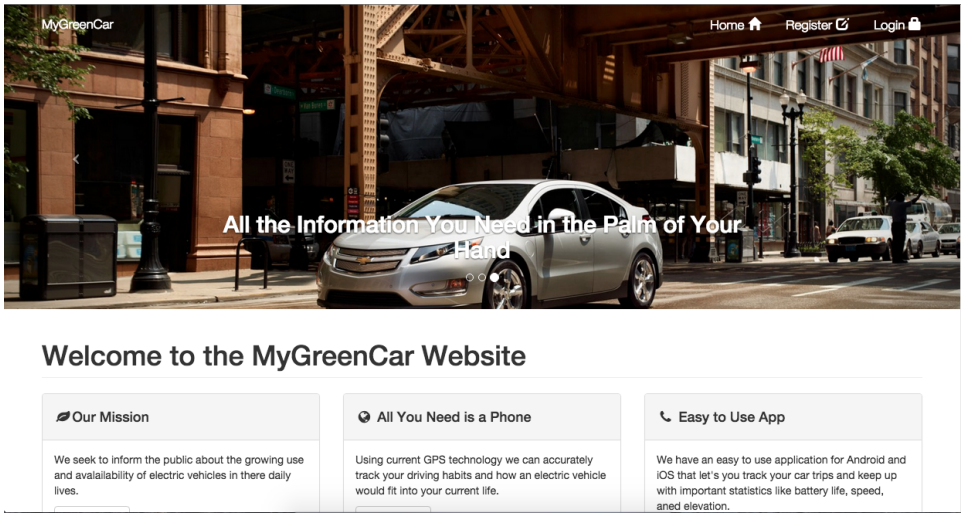


Figure 2.6: MyGreenCar home page

[8] source: `http://www.toptal.com/angular-js/a-step-by-step-guide-to-your-first-angularjs-app`

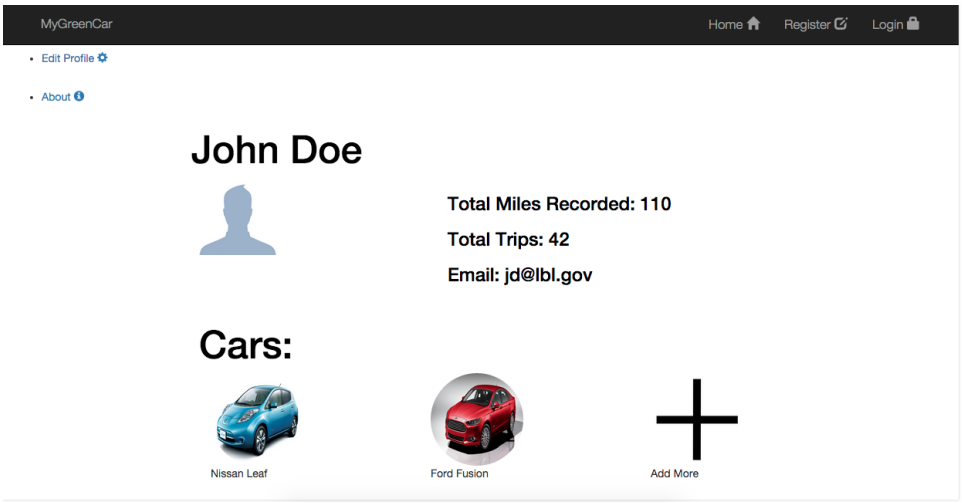


Figure 2.7: MyGreenCar user profile page

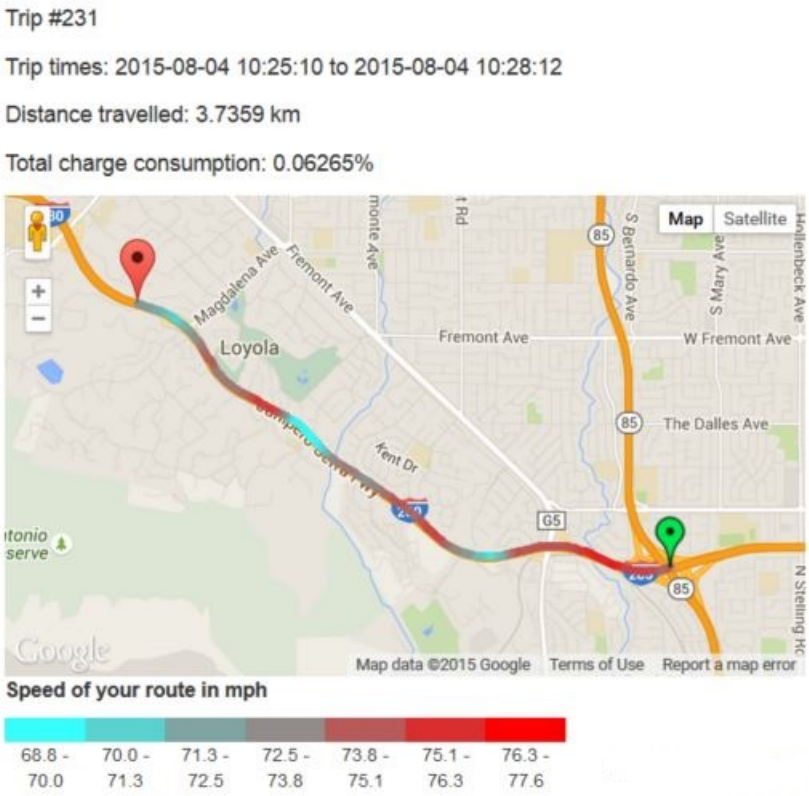


Figure 2.8: Trip overview in the user profile

2.2.2 Vehicle simulation

The vehicle models referred as the power-train models are the core of MyGreenCar within which two different process happen, filtering the raw input data and computing the corresponding consumption. Since my arrival in the team this section had several mutations.

1. V2G-Sim based with simple EV power-train model
2. V2G-Sim based with detailed EV model (accounting for vehicle physics and terrain)
3. new system incorporating compiled detailed power-train model of EV, hybrid and conventional vehicles

Since the last version, detailed power-train models are provided through Simulink's Autonomie model from Argonne National Laboratory[9]. As Simulink models would have been too slow, we use a compiled version of those on the production server. One of our next challenge is to generate enough models to follow the automobile market.

Another challenge of MyGreenCar's team is to cope for the missing raw data values and the lack of precision. Before computation, and especially on terrain data, filters must be applied. More efficient and clever filters could reduce error on the final consumption.

How did we generate fast processing power-train models ?

The answer is: using Rapid Simulation target (RSIM). The Real-Time Workshop Rapid Simulation Target (`RSIM`) consists of a set of target files for nonreal-time execution on your host computer. You can use `RSIM` to generate fast, stand-alone simulations that allow batch parameter tuning and loading of new simulation data from a standard MATLAB MAT-file without needing to recompile your model.

After building an `RSIM` executable you can perform any combination of the following by using command line options. Without recompiling, the rapid simulation target allows you to:

- Specify a new file(s) that provides input signals for From File blocks "`model -f old.mat=new.mat`"
- Replace the entire block diagram parameter vector and run a simulation (not compatible with Autonomie's models)
- Specify a new stop time for ending the stand-alone simulation "`model -s <stoptime>`"

[9] `http://www.autonomie.net/`

- Specify a new name of the MAT-file used to save model output data `"model -t old.mat=new.mat"`

You can run these options:

- Directly from your operating system command line
- By using the bang (!) command in MATLAB prompt

Therefore, you can easily write simple scripts that will run a set of simulations in sequence while using new data sets. These scripts can be written to provide unique filenames for both input parameters and input signals, as well as output filenames for the entire model or for To File blocks. The Figure 2.9 shows a comparison between the Simulink and the compiled model produced, the latter running 10 times faster and producing the same results. *Reference:* `http://radio.feld.cvut.cz/matlab/toolbox/rtw/rtw_ug/rsim.html`

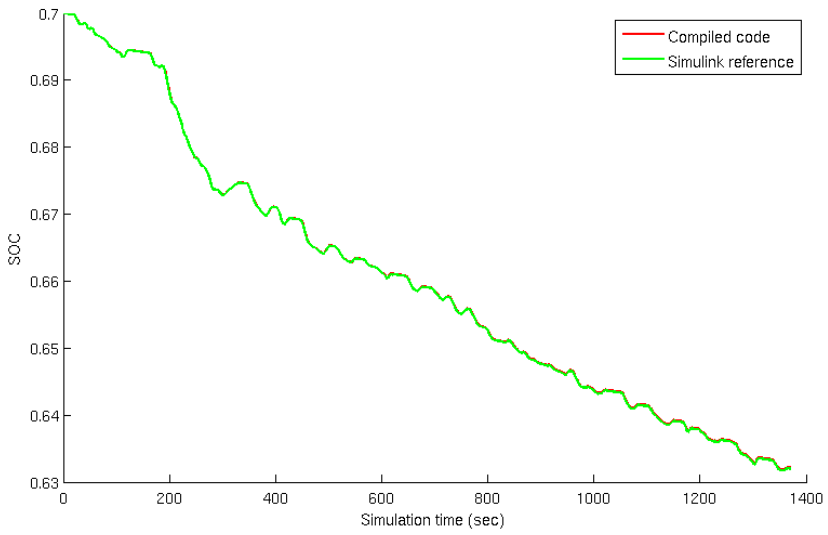


Figure 2.9: EV model comparison on a UDDS drive cycle

2.2.3 Smartphone and web applications

MyGreenCar has three faces: one Android application, one Iphone application and a web application. Both smartphone applications are meant to be end user's sensors; their main goal is to record user's trips. However, people are more and more used to doing everything with their phone, and so our applications also intend to display trip results. Developing one application means developing the other one in the same manner, resulting in a lot of work.

On the other hand the web application is fairly straight forward to develop and is the most accessible of the three applications. In our view the web application will serve as an advance tool to visualize meaning full results. The web site will host different features, from which an easy to use "User Profile" for a quick visualization, but also an in depth MyGreenCar laboratory. The latter will enable people to play with their data in many ways:

- generate their SOC profile versus any range of time (days, week, month)
- compare their consumption with a vast range of vehicles
- find the most fuel efficient road from A to B given traffic, terrain, ...
- display charging stations and strategy to minimize travel cost
- share they travel pattern with other users to find alternatives

The smartphone applications are built to provide the same sort of experience regarding Android or Iphone. The Figures 2.10 and 2.11 are showing the Android and the Iphone version of the "New trip" page. I won't describe here how applications are working internally as it hasn't been my focus during this internship. However I believe that many examples are available on the web (`https://code.google.com/p/open-gpstracker/`) and that the app in its simpler form shouldn't be an overwhelming project for someone familiar with java language.

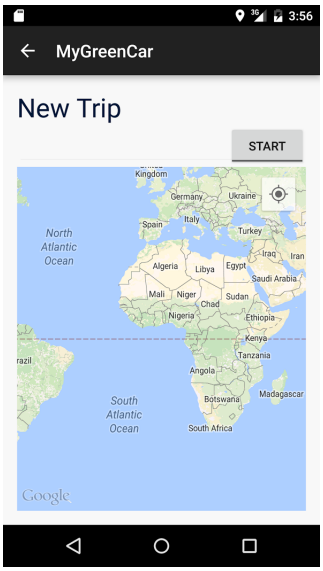


Figure 2.10: Start a new trip under Android

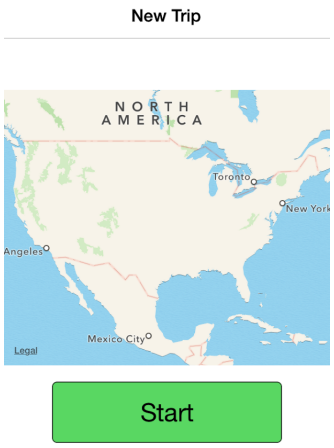


Figure 2.11: Start a new trip under Iphone

2.2.4 MyGreenCar version controlling

This last subsection is not describing a part of MyGreenCar system, but is nevertheless necessary. We are using Git and more precisely GitHub to host our MyGreenCar project (as well as all the other initiative we have). Here is an overview of the Git practises someone needs to follow:

"The production server will maintain a production version of the website, mygreencar-01.lbl.gov, that is always fully operational and totally vetted. The development server contains the development version, mygreencar-ds.lbl.gov, which is where all changes are made and tested. One must never modify the production server or make changes to it; receive changes from the development server only when they have been totally tested. Both sites use the Git repo MyGreenCar-Server."

Here is some basic practises one can follow:

1. Make sure you have pulled the last state of the the branch you are developping on

```
git pull origin MY_BRANCH
```

2. Once you have improved something (and you are happy about it). Add and commit you changes with a nice informative message

```
git add -A
git commit -m "My useful message"
```

3. You have coded all day long committing every hours, push your improvements on-line

```
git push origin MY_BRANCH
```

4. To develop a new feature, fork your project

```
git branch MY_NEW_BRANCH
git checkout MY_NEW_BRANCH
```

5. pull, add, commit and push the same way you did before

6. when it's time merge your branch on the develop branch

```
git checkout develop
git merge MY_NEW_BRANCH
```

7. You are now an accomplished git user ! (but still you might want to take some time exploring the flag options at your disposal)

2.3 MyGreenCar conclusion

As to conclude, MyGreenCar is undoubtedly a very enriching experience. From Simulink compiled model to web development and version controlling I have just learnt so much !

Hopefully it has a bright future in replacing `https://www.fueleconomy.gov/` (milions of users). But before all, it is for me an incredible source of data for mobility analysis purpose and electrical load demand across the city.

3

V2G-Sim

3.1 Introduction

In opposition to the MyGreenCar project V2G-Sim intends to simulate millions of vehicles.

The Vehicle-to-Grid Simulator (V2G-Sim) provides systematic quantitative methods to address the uncertainties and barriers facing vehicle-grid integration (VGI). In the real world, each person drives a different vehicle, in different ways, with different trip distances, at different times. Predicting the adequacy of plug-in electric vehicles (PEVs) for the needs of drivers, and accurately predicting the impacts and opportunities to the electricity grid from increased PEV deployment requires models that can consider these differences at the individual vehicle level.

V2G-Sim models the driving and charging behavior of individual PEVs to generate temporally- and spatially-resolved predictions of grid impacts and opportunities from increased plug-in electric vehicle (PEV) deployment. V2G-Sim provides bottom up modeling from individual vehicle dynamics all the way up to aggregate grid impacts and opportunities.

First will be discussed the technical concepts of V2G-Sim in the MATLAB version. Then the reasons that have lead to create a Python version of V2G-Sim will be discussed. In a third section we will expose the on going research around V2G-Sim.

3.2 V2G-Sim - MATLAB version

At first the V2G-Sim was composed of several MATLAB scripts written by different authors. Later on a Graphical User Interface (GUI) has been developed to create a more stable structure, by gathering and organising functionalities around a common ground. This structure, manifested by the GUI, has for purpose to host algorithms used for simulations. Finally, among other benefits the GUI provides a user-friendly interface which helps

to get started.

V2G-Sim philosophy is to keep a flexible architecture. Every piece is a module, which can be easily replaced or tested. The file hierarchy is representative of different modules, using folders for major V2G-Sim functionality (Figure 3.1).

In order to gain a better understanding, one user needs to understand the major steps in using V2G-Sim:

1. Itineraries generation
2. Model parametrization
3. Result post-processing

The structure of this section will follow the above mentioned list. It should also be noted that the V2G-Sim home page links those three parts together.

3.2.1 Creating itineraries

The term "itinerary" refers to a detailed plan of a journey for one car. A car's journey is also described by activities corresponding to two different states:

- driving
- parked

An itinerary is composed of single activities, represented by eight fields (Table 3.1). Where "Charger" accounts for the power rate at the charging station, and "NHTS weight" accounts for the replicate weight (more details page 26).

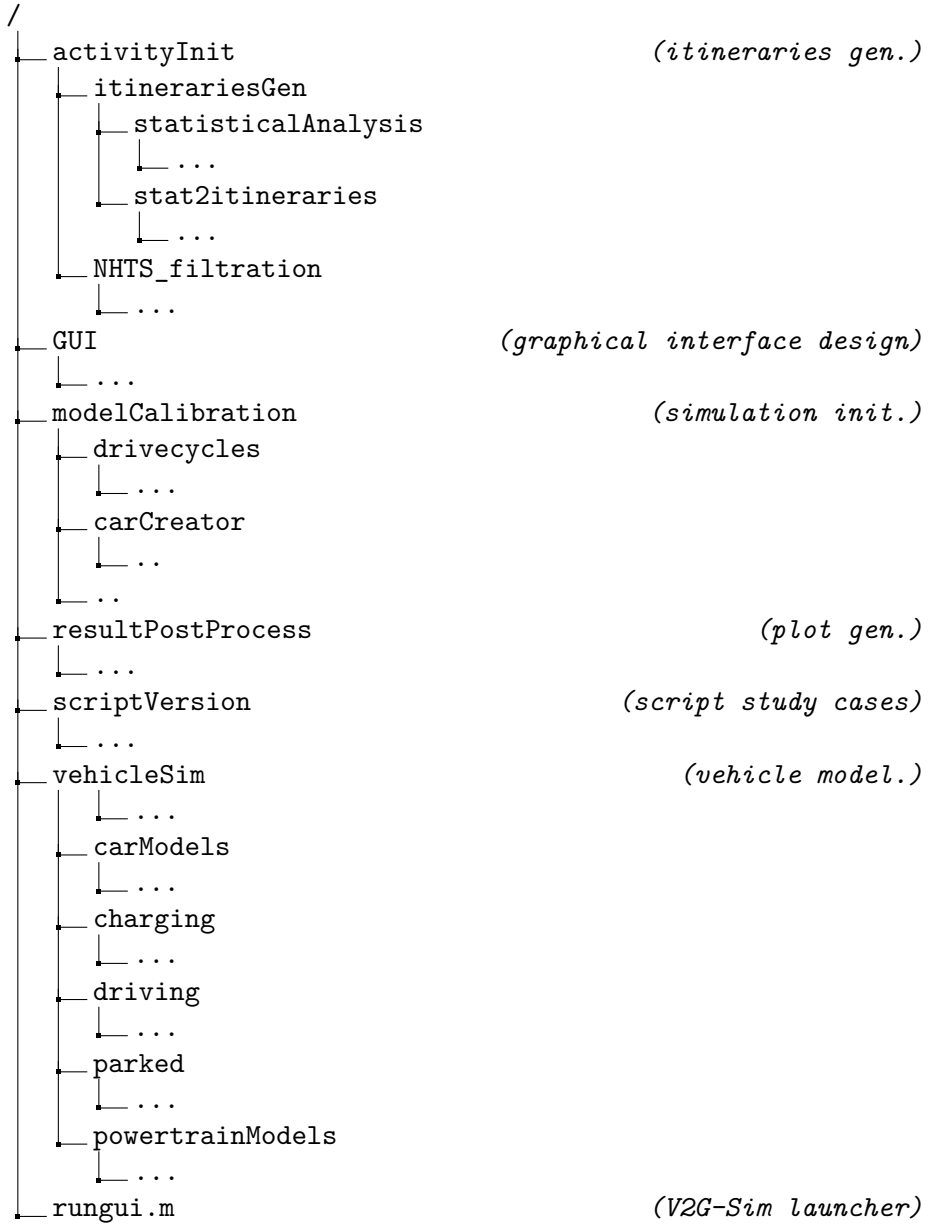
Table 3.1: Activity format

Vehicle ID	Start date (hours)	End date (hours)	Distance (miles)	State	Location	Charger (W)	NHTS weight
1	0	8.5	-1	Parked	Home	1440	193
1	8.5	9	20	Driving	-1	-1	193
1	9	18	-1	Parked	Work	-1	193
1	18	18.5	20	Driving	-1	-1	193
1	18.5	24	-1	Parked	Home	1440	193

V2G-Sim provides two different approaches to create itineraries:

- using real data
- using statistical model

Figure 3.1: V2G-Sim folder hierarchy



The specificities of both approaches are further discussed in the next subsections. The main reason for a statistical approach is to overpass the limited sample of cars in one's input data.

3.2.2 Measured data

When choosing to use actual data, one will be redirected toward the NHTS (National Household Travel Survey) filtering page of V2G-Sim. You can then change the default filtering options in order to generate relevant itineraries. For more information about the 2009 NHTS data, `http://nhts.ornl.gov/`.

It should be noted that "location" in the resulting itinerary file should be understood in a broad sense ("what kind of location") for example, "home" or "work". The number of car resulting in one itinerary file will correspond to the remaining number of cars, after filtering data according to the criteria selected. However, one might want to scale the number of cars simulated. For this purpose, one can use a statistical approach with the data obtained in this section (refer to the next subsection).

Note that in future versions of V2G-Sim, a connection with the growing MyGreenCar database (more than 10 000 miles) will lead to very interesting possibilities, where the "location" could be GPS coordinates.

3.2.3 Statistical model

The statistical approach intends to modify measured data sets and scaling the scope of a study by adding or reducing the number of cars. The approach can also account for creating itineraries from scratch by setting parameters in a statistical analysis.

From real data to a statistical analysis

The input data will be processed into 20 separate bins according to the itinerary of a car during the day. The 20 bins (Table 3.2) capture most of the basic daily transits. The percentage of none-captured car will be displayed on the processing window.

Once the daily transit have been shared into bins, logistic and exponential distributions are parametrised to fit specificities of the bins. Each activity is represented with several distributions to determine:

- a duration
- a distance (for driving)

Tables 3.3 and 3.4 show the structure of the bins #1 and #2 as examples. It should be noted that any number in the distribution column refers to the row index where the value has already been calculated.

Table 3.2: Itinerary captured by each bin

Bin#	Bin mobility pattern
1	home → work → home
2	home → work → lunch → work → home
3	home → work → lunch → work → home
4	home → work → after work destination → home
5	home → work → lunch → work → after work destination → home
6	home → work → home → after work destination → home
7	home → work → lunch → work → home → destination → home
8	home → destination → work → destination → destination → home
9	home → work → destination → destination → home
10	home → destination → work → home → destination → home
11	home → work → home → destination → destination → home
12	home → work → destination → home → destination → home
13	home → work → home → 3 × destination → home
14	home → destination → work → destination → home
15	home → destination → home
16	home → destination1 → destination 2 → home
17	home → destination → home → destination → home
18	home → destination 1 → destination 2 → destination 3 → home
19	home → 4 × destination → home
20	home → 5 × destination → home

A statistical file is composed of 21 separate sheets, with the first summarizing the number of cars in each bin. The following 20 sheets represent the parameters for each bin as shown on Tables 3.3 and 3.4. A simple way to create new itineraries is to modify the number of cars in each bin, as well as the parameters of the distribution representing starting times, durations and distances.

3.2.4 Setting parameters for Simulations

Itineraries correspond to the input data that answer "where and when" are the cars. Before launching a simulation one might as well set the model's options. The model options refer to the following subsections:

- driving model
- car model
- drive cycle generation
- charging model

Table 3.3: Itinerary bin [Home, Work, Home]

Feature	Distribution	Paramter 1	Parameter 2
home	logistic	8.5	1
distance to work	logistic	10	5
work duration	logistic	9	2
distance to home	2	0	0

Table 3.4: Itinerary bin [Home, Work, Lunch, Work, Home]

Feature	Distribution	Paramter 1	Parameter 2
home	logistic	8.5	1
distance to work	logistic	10	5
work duration	logistic	5	1
distance to lunch	exp	2	1
lunch duration	exp	1	0.5
distance to lunch	4	0	0
work duration	logistic	4	1
distance to home	2	0	0

- charger options
- advanced functions

Driving model

The driving model is the module in V2G-Sim where the vehicle consumption is processed. The detailed power-train model accounts for an accurate simulation of EV vehicles, however the computational demands of such model is very high. In order to launch a study case with a large number of vehicles, a less accurate but faster power-train model is available. The detailed power-train model is a MATLAB script based on an Autonomie EV model (`http://www.autonomie.net/`). In contrast the simple power-train model associates consumption values to specific driving speed. When going from a detailed to a simple power-train model, a different car initialization file is needed. As described in the next subsection, the detailed power-train model can be use to parametrise a simple power-train car model.

Car model

The car model determines the characteristics of a single vehicle. When choosing a car model using the interface, the same model is then applied to every car. However, it is possible to specify a model for each car by

opening the "Initialization" sheet inside the itinerary file. A simple power-train model of the car is a set of consumption values under certain drive cycle types (city, highway). In contrast a car model for detailed power-train model is a set of parameters upon mechanical and electrical characteristics.

Drivecycle generation

Drive cycles are generated for each driving state inside one itinerary file. These cycles represent the way cars are driven as well as the terrain changes.

Charging model

The charging model determines the power rate at which a car will charge, this is usually the place where user tend to create their own functions. However three default function are built in the system. The first option allows a car to charge as soon as it is plugged in and until its battery is full. In other words, it is simulating an uncontrolled behavior.

In the second option, the user is able to specify a Demand Response event (DR) at a specific time of the day. The parameters include the DR event duration, the time range within which cars can resume their charging to avoid peak demand, and a minimum SOC buffer available at any time. The algorithm will stop the charging of any vehicles within the time range of the DR event, if the SOC needs are not compromised during the rest of the day.

In the third option, the user can impose a charging rate signal. The signal is followed by a car if its SOC value is above the specified threshold.

Charger options

The charger option searches for all the different locations and their corresponding charging rate inside one's input itineraries. The user may then change the charger type available at one location.

Advanced functions

Our final feature is a way for any user to easily short-circuit the main modules of V2G-Sim. By selecting this option the user can specify an alternative function to use.

3.2.5 Launching a simulation

The last step before launching a simulation is to set the computational parameters. The number of iterations correspond to the number of simulations needed to initialize SOC values for each car. Finally, "Stranding State of Charge" is the minimum SOC value for which V2G-Sim consider a vehicle as stranded.

3.2.6 V2G-Sim batch-processing mode

The graphical user interface is a great tool to learn how the V2G-Sim is working. However it will never reach the flexibility brought by the batch processing mode. In order to exploit the full potential of V2G-Sim, the user can call the V2G-Sim functions within its own script. This functionality enable a user to launch huge batch processing script in order to assess combinations of different parameters (charging algorithm, charging infrastructure, itineraries, car model, drive cycles, ...).

A practical example is structured so that one could re-use it as a template for further research work. The structure is as follow:

1. set a path for the V2G-Sim and input itineraries
2. initialize SOC values and create drive cycles
3. create case studies
4. launch simulations
5. post-process results

3.3 V2G-Sim development, moving to Python ?

Since we have developed compiled model to replace the Simulink car model, we have gained independence, as MATLAB is now avoidable. Here are a few though about MATLAB and Python that explain why we decided to use Python (source: `https://www.stat.washington.edu/~hoytak/blog/whypython.html`)

"MATLAB's language design is focused on matrix and linear algebra operations; for turning such equations into one-liners, it is pretty much unsurpassed. However, move beyond these operations and it often becomes an exercise in frustration."

[...] "The extensive use of Python in everything from system administration and website design to numerical number-crunching shows that it has, indeed, hit the sweet spot. In fact, I've anecdotally observed that becoming better at R leads to skill at interacting with data, becoming better at MATLAB leads to skill at quick-and-dirty scripting, but becoming better at Python leads to genuine programming skill."

There are a few other points worth mentioning:

- readability - "the joke is that turning pseudocode into correct Python code is a matter of correct indentation."
- balance of high level and low level programming - "A general rule of thumb is that your program spends 80% of its time running 20%

of the code. Thus a good strategy for efficient coding is to write everything, profile your code, and optimize the parts that need it. Python's profilers are great, and Cython allows you to do the latter step with minimal effort.

For a 1000 X 1000 array, on my 2.4 GHz laptop, the Python version takes 1.67 seconds, while the Cython version takes only 3.67 milliseconds (a vectorized version of the above using an outer product took 15.1 ms)."

- language interoperability - "As a side affect of its universality, Python excels at gluing other languages together. One can call MATLAB functions from Python (through the MATLAB engine) using MLabWrap."

- hierarchical module system - "Each Python file defines a module. Classes, functions, or variables that are defined in or imported into that file show up in that module's namespace."

- data structures - "While proof-of-concept code often doesn't need optimal data structures, such code causes problems when used in production. Python has lists, tuples, sets, dictionaries, strings, threadsafe queues, and many other types built-in."

- available libraries - "In short, if I want to take a break from writing a bunch of matrix manipulation code and automate an operating system task, I don't have to switch languages."

- testing framework - "Python provides a built-in, low barrier-to-entry testing framework that encourages good test coverage by making the fastest workflow, including debugging time, involve writing test cases."

- Finally, it is completely free and cross platform.

As shown on Figure 3.2 a vehicle has an itinerary which is a set of activities. A vehicle is also associated to a car model, which is a car representation that can take many different values. Each activity has an energy cost, whether it is a driving or parked activity. While driving activities contain information about a drive cycle, parked activities are linked to a location from where they could draw power.

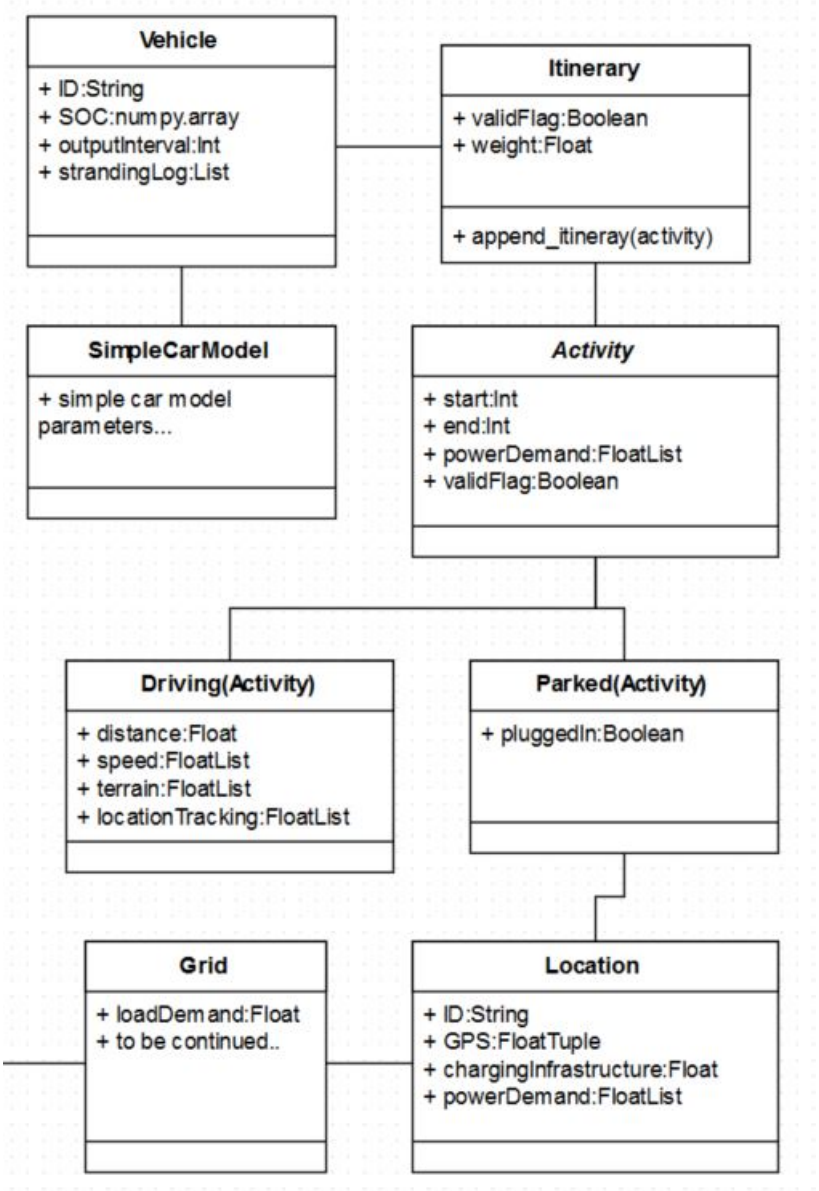


Figure 3.2: UML model of the data structure in V2G-Sim

The following script shows how simple one can interact with the modules once the correct functions are implemented.

```
1 import sys, os
2 sys.path.append(os.path.join(os.path.dirname(__file__),"../
      v2g_sim"))
3 import runV2GSim
4 import postProcess
5 import itinerary
6
```

```
# Initialize location and vehicles from excel file
excelFilename = '/home/jonathan/workspace/Itineraries.xlsx'
vehicleList, locationList = itinerary.load_itinerary_from_excel(
    excelFilename)

# Supress vehicle that could not be initialized correctly
itinerary.suppress_vehicle_with_empty_itinerary(vehicleList)

# Calculate the vehicle SOC and power demand along the day
raw_input("Press Enter to continue...")
runV2GSim.runV2GSim(vehicleList, nbIteration=5)

# Plot results
postProcess.plot_total_EV_power_demand(vehicleList)
postProcess.plot_all_vehicle_SOC(vehicleList)
postProcess.plot_average_SOC(vehicleList)
```

3.4 Research work

First, here is a list of publication made using V2G-Sim:

- S. Saxena, J. MacDonald, and S. Moura, Charging Ahead on the Transition to Electric Vehicles with Standard 120 V Wall Outlets, Applied Energy, in press, 2015.

- S. Saxena, C. Le Floch, J. MacDonald, S. Moura, Quantifying EV Battery End-of-Life through Analysis of Travel Needs with Vehicle Powertrain Models, Journal of Power Sources, 2015, doi: 10.1016/j.jpowsour.2015.01.072.

- S. Saxena, J. MacDonald, D. Black, S. Kiliccote, Quantifying the Flexibility of Electric Vehicles to Avoid Charging during Peak Demand Periods, SAE World Congress, 2015, SAE 2015-01-0304, doi: 10.4271/2015-01-0304.

- T. Markel, A. Meintz, K. Hardy, B. Chen, T, Bohn, J. Smart, D. Scoffield, R. Hovsapian, S. Saxena, J. MacDonald, S. Kiliccote, K. Kahl, R. Pratt, Multi-Lab EV Smart Grid Integration Requirements Study: Providing Guidance on Technology Development and Demonstration, National Renewable Energy Laboratory technical report TP-5400-63963, 2015. `http://www.nrel.gov/docs/fy15osti/63963.pdf`

- C. Le Floch, F.W.M Belletti, S. Saxena, A. Bayen, S. Moura, Distributed Optimal Charging of Electric Vehicles for Demand Response and Load Shaping, accepted for 54th IEEE Conference on Decision and Control, 2015.

Itinerary generation

Itinerary generation is a key feature for V2G analysis. In the recent development of the Python version of V2G-Sim the itinerary combinations are no longer a fix step of bins, but are adapted to the itinerary in the input data. The statistical distribution (logistic and exponential) representing duration and distance of each activity are also topic for research in order to find the distribution having the best fit.

In the next development of V2G-Sim several projects are leading to potential research of mobility modelisation:

- Machine learning algorithm - A trip chain generation method is developed based on the Naive Bayes (Work of Dai Wang) model[1]. It is able to predict the next destination based on the previous one and so forth for a day.

- Database approach - MyGreenCar data could be restricted for an area and serve as highly realistic data for any project

- Multi-Agent Transport Simulation (MATSIM) - With MATSIM the goal is to enable any mobility project to potentially be used to quantify EVs impacts

Integration of renewable energy

In the new V2G-Sim version a section is dedicated to grid data. Modules are being built so that one user can enter historical load data, but also solar irradiation and wind speed data to quantify renewable input. V2G-Sim also envisioned to explore more grid oriented modeling using co-simulation tool (being develop at Berkeley Lab), but also a direct integration of GridLab-D (developed at the Pacific Northwest National Laboratory) in the next steps.

Optimized charging algorithms

The V2G-Sim interface offer the possibility to anyone to easily implement a charging algorithm. Many functions have been implemented to test the flexibility of EVs storage. As the grid data is more and more part of V2G-Sim new charging algorithm explore solution to optimally schedule Plug-in Electric Vehicles charging, with the objective of flattening the net load curve. This leads to a dual maximization problem that can be solved in an iterative and decentralized manner: at each iteration, PEVs solve their optimal problem, communicate their response to the aggregator, which then updates a price signal. (Work of Caroline Le Floch)

[1] `https://en.wikipedia.org/wiki/Naive_Bayes_classifier`

Battery degradation

The concerns about battery degradation is one of the main obstacle to the V2G concept. As V2G-Sim can also takes into account the detailed dynamics of a vehicle, battery degradation is possible to quantify. The Figure 3.3 refer to the structure being implemented in V2G-Sim to capture battery degradation. The user will soon be able to test the performance of a charging algorithm regarding to the battery degradation induced.

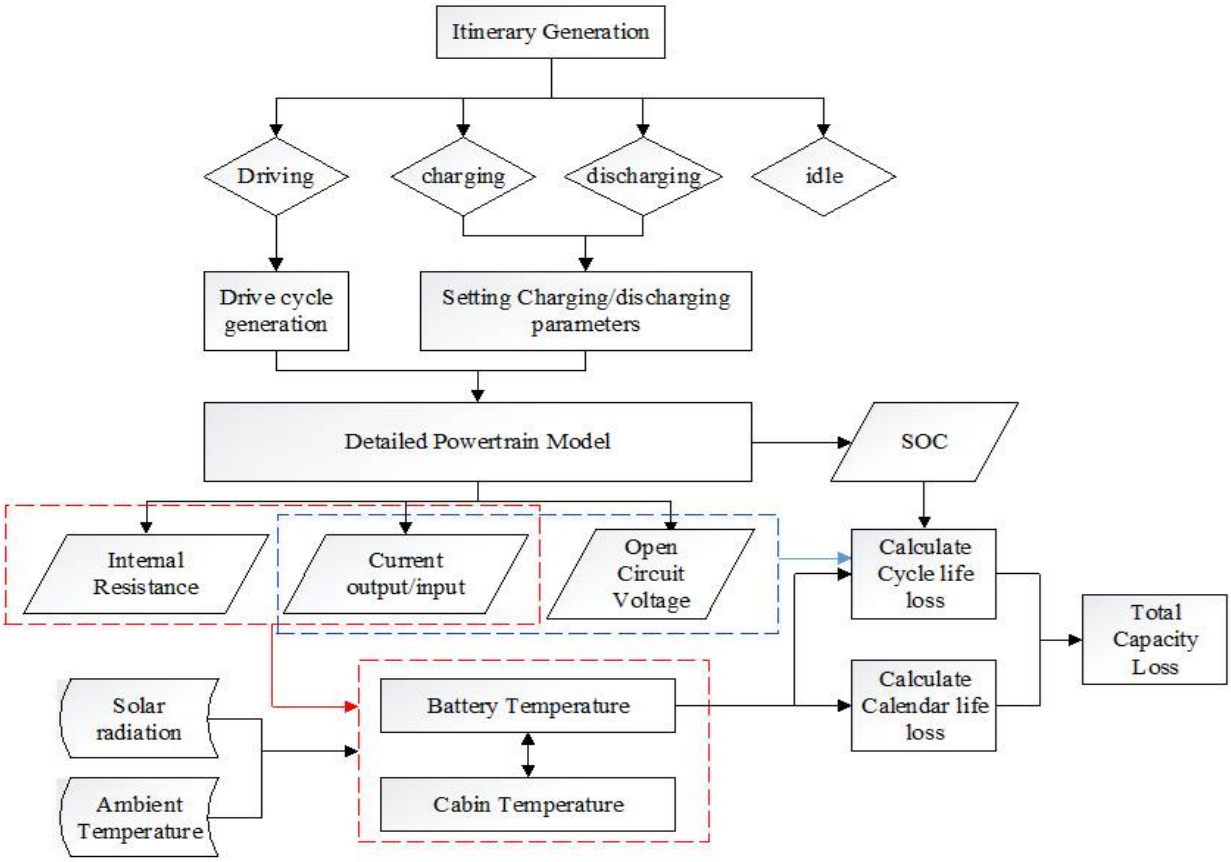


Figure 3.3: Battery degradation model overview

3.5 Conclusion

Since V2G-Sim is becoming an open source Python code, this project is by far the most exciting work I have pursue. We envisioned great coupling between recognize mobility pattern simulator (MATSIM) and grid simulator (PowerFactory, GridLab-D). The V2G-Sim project is for me a good fit. As it is in itself a good description of my formation across urban systems and electrical engineering.

4

Personal experience

As the Berkeley Lab gather researchers around many research topics, I had a very rich experience in my time being at the Lab. Beside my project with Dr Saxena, I have worked with Dr Spyros Chatzivasileiadis on the VirGIL project (Virtual Grid Integration Laboratory). The VirGIL project is a modular co-simulation platform designed to study interactions between demand response strategies, building comfort, communication networks and power system operation. In this project, I have been working on modeling the IEEE 13-bus under PowerFactory, and more recently on the communication between VirGIL and PowerFactory using the FMI standard.

The public talks on the lunch time are also a good way to learn from everybody's work across the lab. For instance, it has been the occasion for me to meet Darren Robinson and read his book "Computer Modelling for Sustainable Urban Design: Physical Principles, Methods and Applications".

Working on MyGreenCar and V2G-Sim projects also brought me to unexpected experience:

- Prepare a 4hours training on V2G-Sim systems for KERI researchers[1].
- Lead a development team of 4 Berkeley students on MyGreenCar project.

4.1 Conclusion

When I first arrived at the Lab, I was determined to work as an electrical engineer on grid related matters. However V2G-Sim and MyGreenCar projects have several faces, including the grid impact of EVs as well as the people's mobility patterns. The latter inspired me a lot, and shifted my topic of interest along this internship. I arrived at the LBNL willing to predominantly work on grid technical realization, I ended up growing interest

[1] `https://www.keri.re.kr/html/en/`

into the urban context. This internship brought me a passion in modeling and simulating different scale urban systems. Hopefully V2G-Sim project will allow me to explore more and more this field with the vehicle perspective. Concerning my next steps, I have been employed at the LBNL as an engineer for the next year, but this doesn't change my desire to pursue a Thesis later on.