2-echelon

Documentation

Author(s): Beatriz Royo

Author’(s’) affiliation (Partner short name): ZLC

Revision History

|  |  |  |
| --- | --- | --- |
| Version No. | Date | Details |
| 1.1 | 21.08.2022 | Second version of the document. |
| 1.2 | 15.09.2022 | Third version of the document |
| 1.3 | 17.10.2022 | Fourth version of the document. |
|  |  |  |
|  |  |  |

Contents

[1 Introduction 4](#_Toc88738680)

[1.1 Scope and objectives 4](#_Toc88738681)

[2 Requirements. 6](#_Toc88738682)

[2.1 Software requirements 6](#_Toc88738683)

[2.2 Input/Outputs 6](#_Toc88738684)

[2.2.1 Inputs 6](#_Toc88738685)

[2.2.2 Outputs 7](#_Toc88738686)

[2.3 Paths structure 7](#_Toc88738687)

[3 Model Description 8](#_Toc88738688)

[4 Instructions to run the model 9](#_Toc88738689)

[4.1 Command line execution of the model 9](#_Toc88738690)

[4.1.1 Instructions and commands 9](#_Toc88738691)

[4.1.2 Arguments 10](#_Toc88738692)

[4.2 Requirements 10](#_Toc88738693)

[4.2.1 Testing requirements 10](#_Toc88738694)

[4.2.2 Folder1 11](#_Toc88738695)

[4.2.3 Folder2 11](#_Toc88738696)

List of tables

[Table 1 2-echelon model – Inputs 6](#_Toc88738697)

# Introduction

## Scope and objectives

The -ecommerce growth has increased the transport flows in the cities while policymakers are introducing delivery constraints with the creation of low emissions zones or low traffic zones that banned the access to fossil fuel or high size vehicles. This panorama is forcing last mile sector to shift from the business as usual (BAU) scenario or 1-echelon (Figure 1 – left) to 2-echelon networks (Figure 1 – right).

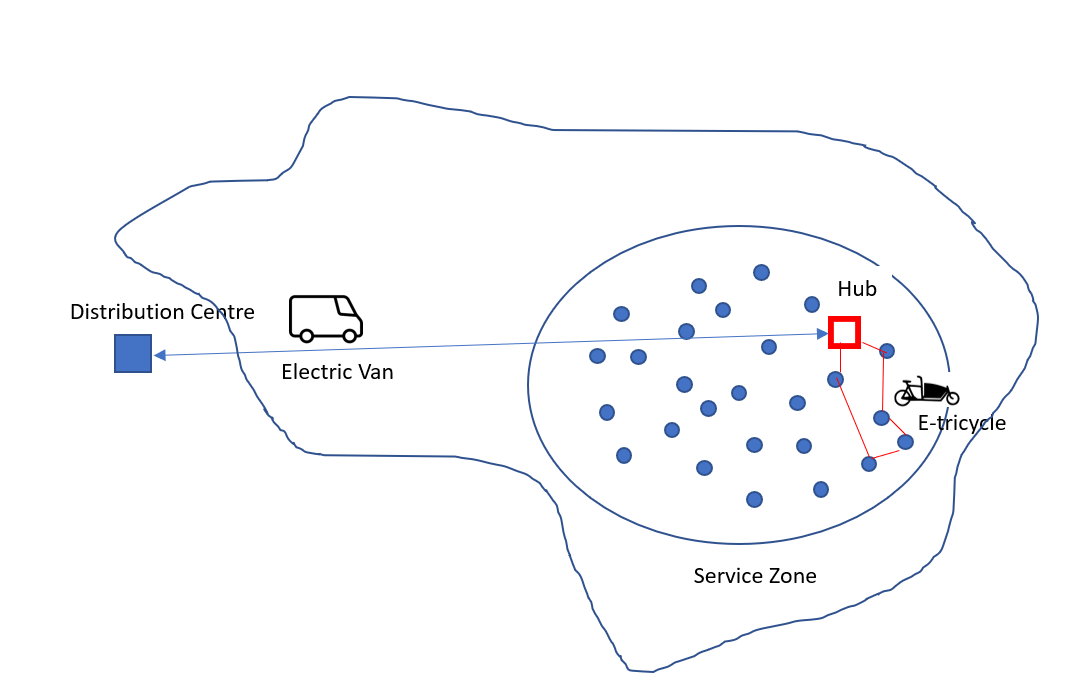
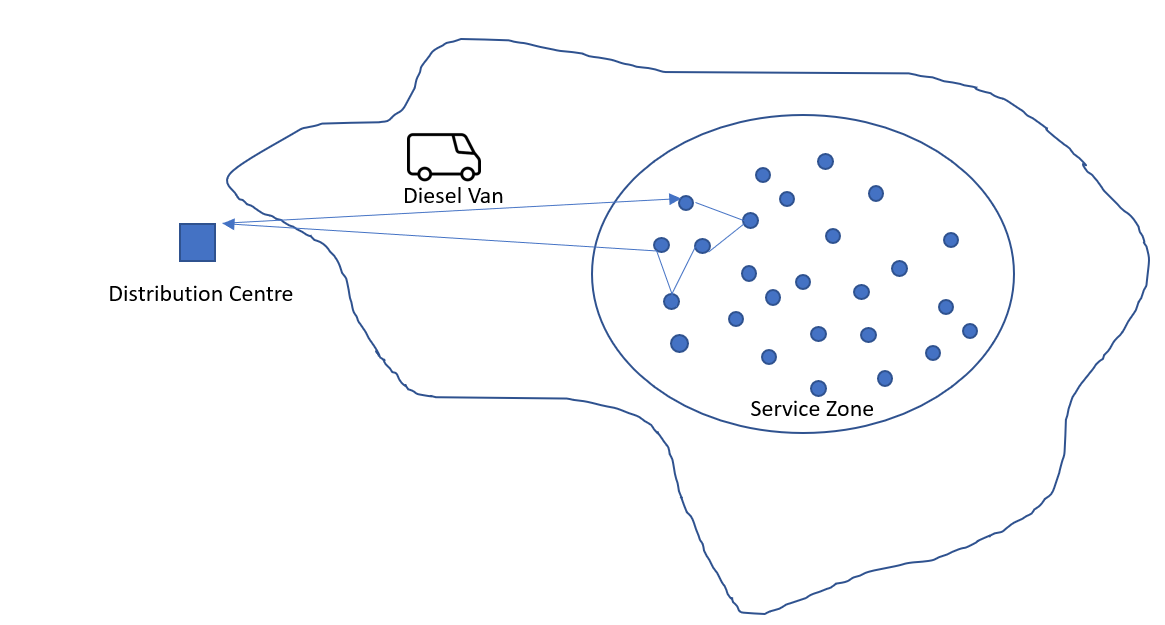


Figure 1. 1-Echelon vs 2- Echelon last-mile distribution networks.

The objective of this model is to support the last-mile sector to compare the impacts of different configuration networks with minimums levels of data granularity. Based on the results (number of vehicles, distances and time) the user can calculate the environmental, operational and financial impact and easily respond to a wide list of “what if” scenarios such as:

* “*what if I move from 1-echelon network using Diesel vans to a 1-echelon using electric vans?”*
* “*what if I move from 1-echelon network using Diesel vans to a 2-echelon using electric vans and e-trikes?”*
* “*what if I install the hub within the city in another location?”*
* *“what if the demand changes”?*
* *….*

Regardless of the scenario, we assume a single distribution Centre (origin) where products are demanded at many destinations within a specified service zone, using a unique type of vehicle with known capacity and within a specific time. We assume all the nodes within the service zone are delivery points.

For estimating the resources needed, the distances and the delivery time per type of vehicle, the following assumptions are required according to the scenario.

1-echelon:

For this scenario (Figure 1 – left) there is only a distribution Centre to serve to the specified service zone. It uses only one type of vehicle “i” with a known capacity. Handling time in the distribution Centre is constant*.*

**Single Distribution Centre**

* Surrounding the city boundaries
* Products are demanded at many destinations
* All destinations within specified service zone
* Hanldling time is considered

**Assumptions**

* 1 echelon network
* Vehicles are homogenous
* Same Capacity
* Workshift constant
* Handling time is constant

2-echelon:

For this scenario (Figure 1 – right), there is a distribution Centre and a hub to serve to the specified service zone. The hub is cross-dock facility where the goods are moved from one vehicle of type “i” supplying the hub from the distribution Centre with direct shipments to another vehicle of type “j” distributing from the hub to the service zone with multiple delivery points. Vehicle type “i” is different to vehicle type “j”. Vehicles type “i" and “j” are known and constant. Handling time in the distribution Centre is constant and handling time in the hub is zero. In the hub we assume the shift time is included in the stop time of vehicle of type “i”. The workshift of the driver of vehicle of type “i” is independent of vehicle type “j” driver’s workshift.

**Single Distribution Centre**

* Surrounding the city boundaries
* Products are for the hub
* All destinations to the hub
* Handling time is considered

**Single crossdock**

* Products from one origin
* Products are demanded at many destinations
* All destinations within specified service zone
* Handling time not considered

**Assumptions**

* 2 echelon network
* Vehicles for 1st leg are homogenous
* Vehicles for 2nd leg are homogenous
* 1st leg and 2ng leg vehicles capacities and characteristics are known
* 1st leg and 2ng leg vehicles are different
* 1st leg and 2ng leg constant and independent
* Handling time for the DC is constant
* Handling time for the hub is zero, shift time include in the stop time of the 1st leg

# Requirements.

## Software requirements

The language and the software version for running the model is shown in Figure 2.:

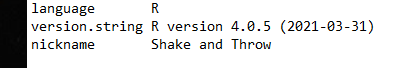


Figure 2. Model version of the language and software.

The following libraries are needed for obtaining the area and centroid of the area of the shapefile with the boundaries of the service area (optional).

* geojosonio package in R ([Cran](https://cran.r-project.org/web/packages/geojsonio/index.html), [github](https://github.com/ropensci/geojsonio))
* sf package in R ([Cran](https://cran.r-project.org/web/packages/sf/index.html) [github](https://github.com/r-spatial/sf)),
* stringr in R ([Cran](https://cran.r-project.org/web/packages/stringr/index.html))
* josnlite in R ([Cran](https://cran.r-project.org/web/packages/jsonlite/index.html))

## Input/Outputs

### Inputs

The 2-echelon model has the following inputs and its outputs.

Table 1 2-echelon model – Inputs

| Inputs | Description |
| --- | --- |
| facilities.csv | Table with the characteristics of the facilities. It will contain 1 row if one-echelon; 2 rows if two-echelons.   * **name**\_ string with the name. * Street: string with the street name. * Number: integer with the number of the street. * Zone: string with the name of the zone. * zipcode: string with the zipcode. * **latitude**: WGS84 latitude (decimal) * **longitude**: WGS84 longitude (decimal). * handlingTime: (hours) to prepare the services. (deprecated: now from zones.csv) * startingTime (time): time the service starts (not used) * endingTime (time): time the service ends (not used) |
| vehicles.csv | Table with the characteristics of the vehicles. It will contain 1 row if one-echelon; 2 rows if two-echelons.   * **Name:** string with the name. * **capacityUnits**: integer capacity in units (eg. parcels) * **capacityKg**: double capacity in kg * xxx: not used * yyy: not used * **velocity**: double (km/h) * autonomy: integer (km) * stopTime: double, average hours for delivering goods in a point within the delivery zone (deprecated now from zones.csv) |
| services.csv | Table with the with the list of orders to deliver within the delivery or service zone. The same file for one-echelon and two-echelon networks. there are 20 columns but only 19 (goods to pick up) and 20 (goods to deliver) are used. Optional. If not available or selected, the model will use the attributes avg\_size and total\_servicies from zones.csv. |
| Config.csv | Configuration file with some parameters and default values. The same file for one-echelon and two-echelon networks.   * **k**: double, a parameter required by Daganzo’s distance. * **workshift** double, hours available for the services for each echelon. * **branchHandlingTime**: double (hours) used by facility1 to prepared the goods. if two echelon is zero. * **UCCHandlingTime**: double (hours) used by facility2 to prepared the goods. if one echelon is zero. * **stopTimeFirstEchelon**: double (hours) required for delivering in the delivery zone. If two-echelon is zero, * **stopTimeSecondEchelon**; double (hours) required for delivering in the delivery zone. If one echelon this value is zero. * **distanceType**: {1 if Euclidean; 2 if Haversine}. 2 by default. * **haversineCalibration**: double, parameter to adjust haversine distance and the google distance (only direct distance) |
| zones.csv | Table with the information about the zone to deliver.   * **id**: integer (identifier) * **Name**: string (name of the zone) * **url\_exist:**{1 if there is some service with a link to download the shapefile; otherwise, 0} * **url**: string with the link to download the shapefile. * **shapefile\_exist:**{1 if shapefile available in shapefile path} * **shapefile\_path**: string with the location of the shapefile. * **area (km2)**: double with km2 of the delivery zone. * **centroid\_latitude\_x**: WGS84 latitude (decimal) * **centroid\_longitude\_y** WGS84 longitude (decimal) * **avg\_size:** double (average size of goods to delivery per delivery point) * **total\_services:** total number of delivery points. |

### Outputs

Table 2. 2-echelon model – Outputs

| Outputs | Description |
| --- | --- |
| output\_two\_echelon.json | Json file with the results. It will contain one record if one-echelon or 2 records if two-echelon.   * **'echelon'** number of the leg {1:2} * **'zone\_name'**=name of the zone of the leg echelon, * **'zone\_avg\_size'**= average size of the item delivered of the zone of the leg echelon, * **'zone\_area\_km2**'=square km of the delivery zone of the zone of the leg echelon, * **'zone\_total\_services'**=number of services to be delivered in zone\_name of the zone of the leg echelon, * **'zone\_latitude'**= coordinate x of some concentric point of zone\_name of the zone of the leg echelon, * **'zone\_longitude'**=coordinate y of some concentric point of zone\_name of the zone of the leg echelon, * **'facility\_name'** = name of the facility of the leg echelon, * **'facility\_handling\_time'** = handling time facility of the leg echelon, * **'facility\_latitude'**= latitude of the facility of the leg echelon, * **'facility\_longitude'**= longitude of the facility of the leg echelon, * '**vehicle\_name'**= name of the vehicle of the leg echelon, * **'vehicle\_capacity**'= capacity of the vehicle of the leg echelon, * **'vehicle\_velocity\_km.s'**= velocity of the vehicle of the leg echelon, * **'vehicle\_velocity\_stop\_time'**= stop time of the vehicle of the leg echelon (h), * **'total\_distance\_km'** = total distance made by the vehicle of the leg echelon to fulfil the leg echelon (km), * **'total\_time\_hours'** = total time made by the vehicle of the leg echelon to fulfil the leg echelon (h), * **'number\_vehicles'** = total distance made by the vehicle of the leg echelon to fulfil the leg echelon, |

## Paths structure

The structure of the folder for running the 2-echelon model is presented in Figure 3.

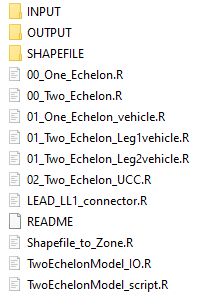


Figure 3. 2-Echelon directory structure.

The INPUT folder contains the files described in section 2.2.1.

The OUTPUT folder is the receipt of the result of executing the 2-echelon model. After the exection of the model, it will contain the json file described in section 2.2.2.

The SHAPEFILE folder contains a shapefile of the delivery zone if it exists. It can be the folder indicated in *‘shapefile\_path*’ of zones in Table 1 or the folder to download the shapefile of the delivery zone indicated in *’url*’ of zones in Table 1.

The scripts for running the scenarios are explained in the following section.

README file contains essential information about the content of the folder and how execute the model.

# Model Description

This section describes the different files and scripts present in the model

| File name | Location | Description |
| --- | --- | --- |
| Shapefile\_to\_Zone.R | Root folder | Script with the functions for reading geographic data, it downloads the shapefiles from url specified in zones in Table 1. |
| TwoEchelonModel\_IO.R | Root folder | functions for reading the input parameters in the folder INPUT and writing the results in the folder OUTPUT folder.It contains complementary functions for converting the output data in the json files used by the wrapper functions to connect to the corresponding models available in the LEAD library (COPERT) and EVCO2. |
| TwoEchelonModel\_script.R | Root folder | Functions for calculating the number of vehicles, distance and times for delivering for one echelon or two echelon configurations. |
| 00\_One\_Echelon.R | Root folder | Script with the example code to run the one echelon model using default data saved in the csv files in the INPUT folder (see 2.2.1). |
| 00\_Two\_Echelon.R | Root folder | Script with the example code to run the two-echelon model using default data saved in the csv files in the INPUT folder (see 2.2.1). |
| 01\_One\_Echelon\_vehicle.R | Root folder | Script with the example code to run the one echelon model using default data saved in the csv files in the INPUT folder (see 2.2.1) but using another vehicle. The parameters are the name of vehicle[[1]](#footnote-1), the capacity, the speed (km/h). |
| 01\_Two\_Echelon\_Leg1vehicle.R | Root folder | Script with the example code to run the two-echelon model using default data saved in the csv files in the INPUT folder (see 2.2.1) but using another vehicle for the first leg. The parameters are the name of vehicle[[2]](#footnote-2), the capacity, the speed (km/h). |
| 01\_Two\_Echelon\_Leg2vehicle.R | Root folder | Script with the example code to run the two-echelon model using default data saved in the csv files in the INPUT folder (see 2.2.1) but using another vehicle for the second leg. The parameters are the name of vehicle[[3]](#footnote-3), the capacity, the speed (km/h). |
| 02\_Two\_Echelon\_UCC.R | Root folder | Script with the example code to run the two-echelon model using default data saved in the csv files in the INPUT folder (see 2.2.1) but using another location for the UCC. The parameters are latitude and longitude (see format in Table 1.) |
| LEAD\_LL1\_connector.R | Root folder | Script for creating the inputs and the dependencies for wrapping function that connects to the next models (COPERT, EVCO2, both) |

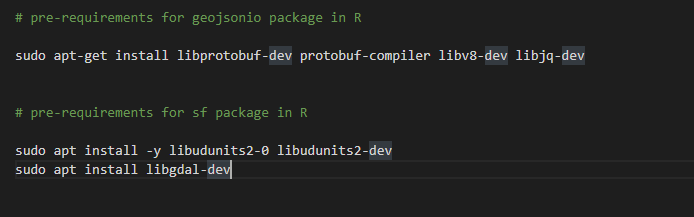


Figure 4. Pre-requirements.

Commands to run the model

1. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/00\_One\_Echelon.R C:/LEAD\_MODELS/NEW/2echelon-main/
2. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/00\_Two\_Echelon.R C:/LEAD\_MODELS/NEW/2echelon-main/
3. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/01\_One\_Echelon\_vehicle.R C:/LEAD\_MODELS/NEW/2echelon-main/ "NV200,Electric,Cargo,Vehicle" 161 45
4. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/01\_Two\_Echelon\_Leg1vehicle.R C:/LEAD\_MODELS/NEW/2echelon-main/ "NV200,Electric,Cargo,Vehicle" 161 45
5. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/01\_Two\_Echelon\_Leg2vehicle.R C:/LEAD\_MODELS/NEW/2echelon-main/ "NV200,Electric,Cargo,Vehicle" 161 45
6. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/02\_Two\_Echelon\_UCC.R C:/LEAD\_MODELS/NEW/2echelon-main/ 40.4161737 -3.7087409
7. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/LEAD\_LL1\_connector.R "\_as-is\_zlc\_electric\_van"

### Arguments

Identification of the arguments introduced in the command line by the scripts above are described below:

1. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/00\_One\_Echelon.R C:/LEAD\_MODELS/NEW/2echelon-main/
2. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/00\_Two\_Echelon.R C:/LEAD\_MODELS/NEW/2echelon-main/
   * args[1] = the location of the folder that contains the scripts, the INPUT, OUTPUT, SHAPEFILES folder to run the scenario
3. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/01\_One\_Echelon\_vehicle.R C:/LEAD\_MODELS/NEW/2echelon-main/ "NV200,Electric,Cargo,Vehicle" 161 45
4. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/01\_Two\_Echelon\_Leg1vehicle.R C:/LEAD\_MODELS/NEW/2echelon-main/ "NV200,Electric,Cargo,Vehicle" 161 45
5. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/01\_Two\_Echelon\_Leg2vehicle.R C:/LEAD\_MODELS/NEW/2echelon-main/ "NV200,Electric,Cargo,Vehicle" 161 45
   * args[1] = the location of the folder that contains the scripts, the INPUT, OUTPUT, SHAPEFILES folder to run the scenario
   * args[2] = the name of the vehicle
   * args[3] = the capacity of the vehicle (same metric that the average service)
   * args[4] = the speed of the vehicle (km/s)
6. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/02\_Two\_Echelon\_UCC.R C:/LEAD\_MODELS/NEW/2echelon-main/ 40.4161737 -3.7087409
   * args[1] = the location of the folder that contains the scripts, the INPUT, OUTPUT, SHAPEFILES folder to run the scenario
   * args[2] = latitude
   * args[3] = longitude
7. Rscript C:/LEAD\_MODELS/NEW/2echelon-main/LEAD\_LL1\_connector.R "\_as-is\_zlc\_electric\_van"
   * args[1] = name of the folder will be created to save the intermediate results will be used by the jupyter notebooks. It must follow the name convention defined with LMT. Possibilities:
     + *[1] name\_folder\_out="\_as-is\_zlc\_diesel\_van" ---> AS IS (script 00\_One\_Echelon.R)*
     + *[2] name\_folder\_out="\_as-is\_zlc\_electric\_van"---> AS IS using electric van (script 01\_One\_Echelon\_vehicle.R with the parameters of the electric van)*
     + *[3] name\_folder\_out="\_as-is\_zlc\_hybrid\_van"---> AS IS using hybrid van (script 01\_One\_Echelon\_vehicle.R with the parameters of the hybrid van))*
     + *[4] name\_folder\_out="\_as-is\_zlc\_electric\_scooter"---> AS IS using hybrid van (script 01\_One\_Echelon\_vehicle.R with the parameters of the electric scooters))*
     + *[5] name\_folder\_out="\_to-be\_zlc\_hybrid\_van \_electric\_scooter"---> TO BE (script 00\_Two\_Echelon.R)*
     + *[6] name\_folder\_out="\_to-be\_zlc\_electric\_van \_electric\_scooter" ---> TO BE using electric van in the first leg (script 01\_Two\_Echelon\_Leg1vehicle.R with the parameters of the electric van)*
     + *[7] name\_folder\_out="\_to-be\_zlc\_hybrid\_van \_electric\_scooter\_newUCC" --> TO BE changing the location of the UCC (02\_Two\_Echelon\_UCC.R with the latitude and longitude of the new UCC)*

## Requirements

### Testing requirements

# System requirements for rjava R library

sudo apt-get install -y default-jre

sudo apt-get install -y default-jdk

sudo R CMD javareconf

# pre-requirements for geojsonio package in R

sudo apt-get install libprotobuf-dev protobuf-compiler libv8-dev libjq-dev

# pre-requirements for sf package in R

sudo apt install -y libudunits2-0 libudunits2-dev

sudo apt install libgdal-dev

### Folder INPUT

The inputs of the model see Table 1.

### Folder OUTPUT

The outputs of the model see Table 2.

### Folder SHAPEFILE

Optional folder to unzip the geographic data of the model if available in an open repository as in the case of the example. The content below is after downloading and unzipping the information of the Low Emissions Zone (LEZ) in Madrid from the open repository.

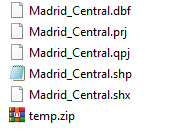


Figure 5. Content of Optional Folder (SHAPEFILE) from the open repository of the LEZ of Madrid

1. The case of the LEAD LL1 Madrid used for testing the model uses a sepecific names convention to connect with the next leve lof models. [↑](#footnote-ref-1)
2. The case of the LEAD LL1 Madrid used for testing the model uses a sepecific names convention to connect with the next leve lof models. [↑](#footnote-ref-2)
3. The case of the LEAD LL1 Madrid used for testing the model uses a sepecific names convention to connect with the next leve lof models. [↑](#footnote-ref-3)