iBRoad

Individual Building
Renovation Roadmaps

Simple energy demand calculation tool

Open source programme code of techno-economic default assessment of renovation measures

TU Wien – Technische Universität Wien

November 2018

www.ibroad-project.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 754045

Authors

Iná Eugenio Noronha Maia, Technische Universität Wien

Lukas Kranzl, Technische Universität Wien

Andreas Müller, e-think

David Schmidinger, Technische Universität Wien

Reviewer

Alexander Deliyannis, Sympraxis Team

Page layout

Sympraxis Team

Cover illustration

depositphotos.com/orelphoto2/smuki / Sympraxis Team

Published in November 2018 by iBRoad.

Table of Contents

I. Introduction 4

II. Programme Code information 6

i. How the code and its calculation procedures are structured 6

ii. How to run the code 9

Step 1 9
Step 2 9
Step 3 (optional) 9
Step 4 10

iii. Where the code is hosted 10

I. INTRODUCTION

The objective and scope of this work under the project iBRoad, is to provide an open source programme code for the techno-economic assessment of renovation measures and alternative renovation pathways of a building. The present outcome includes the open source programme code, written in the programming language Python, of a simplified standardised energy demand model to assess useful, final and primary energy demand of residential houses. The model consists of a monthly-based energy balance calculation based on calculation procedures suggested by the German DIN-Standards 18599. The economic part of the assessment will be delivered at a further project phase.

The present outcome consists of two parts, this report (first part), with the documentation and explanation of the programme code (second part). The intention of this report is to give a short explanation about how this code is structured, presenting the main calculation procedures written in the code files, and therefore the necessary input parameters, allowing the code user to run the calculation his/herself and obtain the results (outputs) of the calculation.

Figure 1 shows the possible integration of the open-source programme code in the iBRoad Concept.

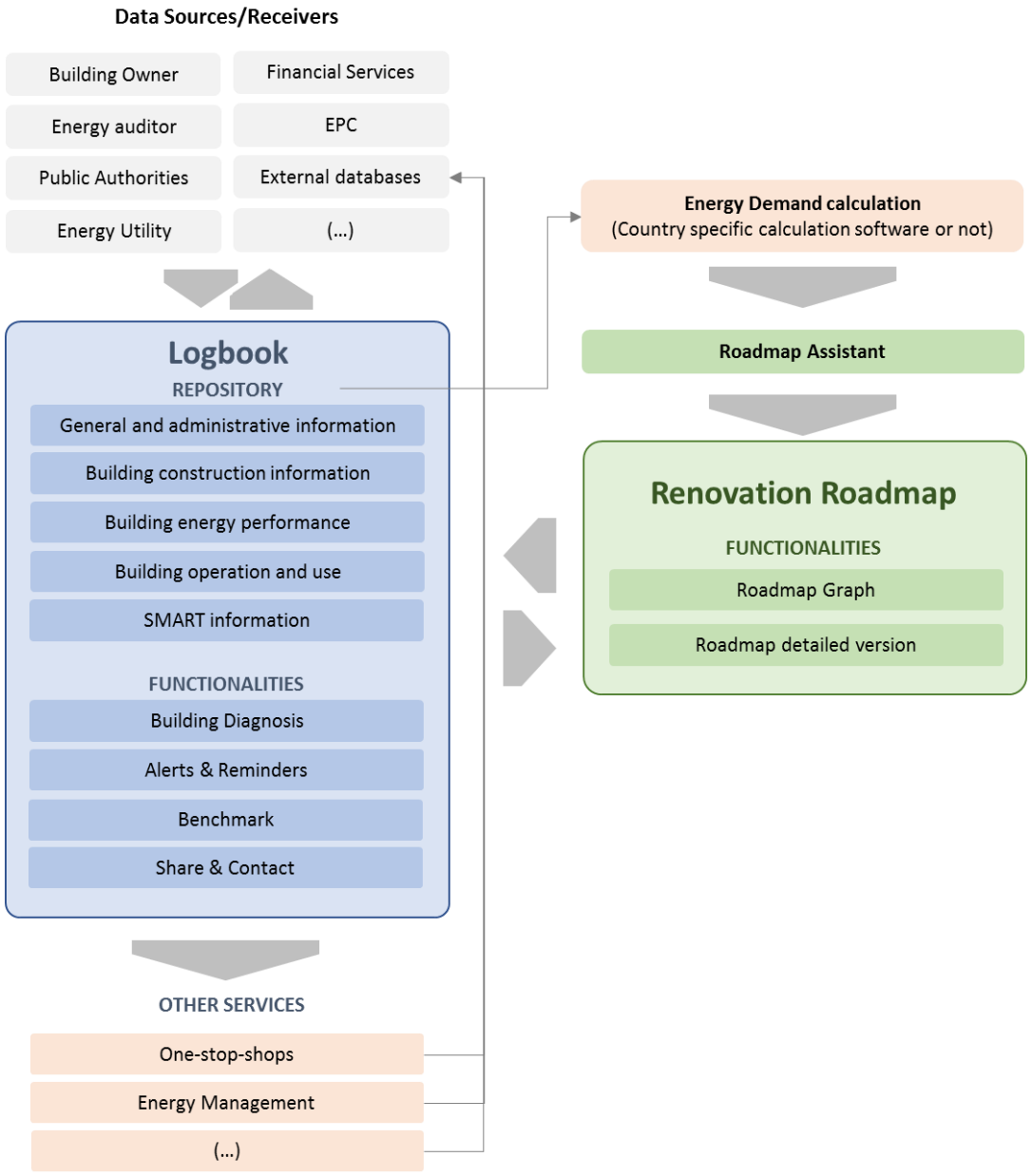

Figure 1: *iBRoad concept overview*

In the iBRoad Concept, the energy demand is calculated with the country specific calculation software or the programme code delivered hereby. The next chapter explains, in more detail, how the open source programme code can be used to calculate the energy demand.

II. PROGRAMME CODE INFORMATION

i. How the code and its calculation procedures are structured

The Open source programme code for energy demand calculation is mainly structured in three parts: input, calculation and output. As illustrated in Figure 2 below, the input information (in green) is divided in different files, where the user provides the main building related information in the file **<building_input.xlsx>** and the other input files are automatically imported into the calculation. The calculation part is also divided in different files (in orange). Here, the user sets the calculation configurations and runs the code in the file **<main.py>**, while the other calculation files run automatically after the "run" command. Finally, the calculation outputs (in blue) are automatically exported to the file **<building_output.xlsx>**, where the user visualises the calculation results.

In brief, with this structure, the code user manages with three main files <building_input.xlsx>, <main.py> and <building_output.xlsx> to calculate the energy demand of the building status quo and the foreseen renovation measures.

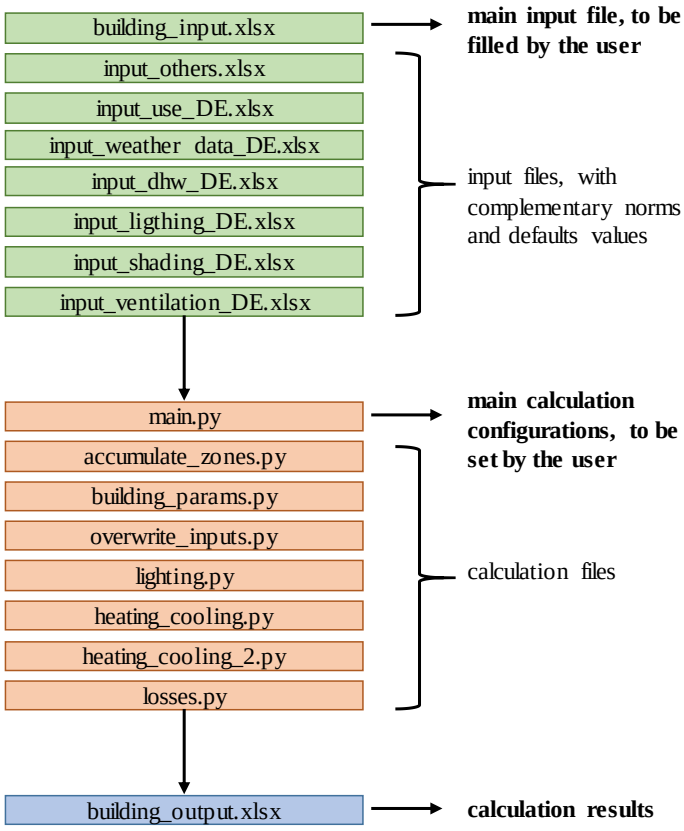

Figure 2: *Programme code overview*

However, this open source programme code allows, firstly a transparent access of the input, as the input files can be visualised and changed by the user; secondly, a modular approach of the energy demand calculation procedures, making the calculation steps clearer. Further on, this concept is described in detail.

As mentioned, the code starts in the file <main.py>, where the user sets the calculation configurations (for more detail, see the section "How to run the code"). By clicking on the command "run", the calculation procedures starts in the main calculation file, <accumulate_zones.py>, which has three sections: 1) input data preparation, 2) calculation scripts and 3) output data definition.

In the first section, the input parameters from the different input files are centralised and prepared. The building input parameters are classified in two types: the "direct parameters", which come from the code user by filling the input file, and the "indirect parameters", which are imported from external files, as exemplary default values for Germany.

The "direct parameters" are defined in the file **<building_input.xlsx>, worksheet "Inputs",** which contains the building input parameters including building geometry, building envelope quality, technical systems characteristics like distribution system, control system, building use profiles and indoor temperatures listed over the lines. The user should define and enter the parameter values in column F. In the other columns, different information about the parameters can be seen: parameter name in German and in English (columns A and B), variable name in German and English (columns C and D), and variable unit (column E), and additional comments to help the code user (column G). For a step-by-step calculation, the new variable values can be entered from column H onwards, according to the number of steps, for example, if the step-by-step renovation plan has 3 steps, then the new variable values are entered in the columns H, I and J (for more details, see the section "How to run the code").

The "indirect parameters" come from seven different input data files, namely:

1. **<input_weather data_DE.xlsx>** contains parameter values for the country specific monthly average solar radiation according to the building orientation and the surface inclination, the monthly average outside air temperature and the maximum hourly solar radiation.

2. **<input_use_DE.xlsx>** contains parameter values for the building use profile and operation as, for example, daily operation hours during the day and the night, operation hours of the heating and cooling systems and others.

3. **<input_lighting_DE.xlsx>** contains parameter values for U- and g-values according to the glazing type, and other factors according to the façade system and sun protection system.

4. **<input_ventilation_DE.xlsx>** contains parameter values for the operation of the mechanical ventilation systems, as set point temperature, monthly average supply temperature of the ventilation systems and others.

5. **<input_shading_DE.xlsx>** contains parameter values for the shading factors, which varies according to the orientation, inclination, season and control strategy and others.

6. **<input_dhw_DE.xlsx>**, similarly to the file <input_use_DE.xlsx>, contains parameter values according to the domestic hot water use profile and operation. The code user has the possibility to substitute these default values.

7. **<input_others.xlsx>**, contains norm, constant, factors (primary energy and CO_2 emissions) values.

In the file **<building_input.xlsx>, worksheet "Rewritable Inputs",** some indirect input parameters (e.g. set indoor temperature for heating, or operation hours the heating system) can be viewed and overwritten, if the user wants to correct some indirect value. This functionality should facilitate the practical handling with the code, by avoiding that the user has to open many files at the same time. This

overwriting procedure is defined in the file **<overwrite_inputs.py>**, and new values are automatically translated to the code, by the command "run" (see also "How to run the code?", Step 3).

The second section of the file **<accumulate_zones.py>** presents the calculation scripts. To execute the calculation procedures, all the building input data is loaded into dictionaries. Then, they pass to the five calculation files, which run sequentially:

1. **<building_params>** prepares the inputted data by doing some pre-calculation of the <building_input>, which will proceed in the next files.

2. **<lighting>** calculation of final energy demand for lighting

3. **<heating_cooling>** calculation of useful energy demand for heating and cooling (monthly based)

4. **<heating_cooling2>** continuation of the file <heating_cooling>

5. **<losses>** calculation distribution losses, use of renewable energy (CHP, PV etc), final and primary energy demand and CO_2 emissions

In general, the code uses the structure of dictionaries to organise and store the inputted, pre-calculated data and output data. Figure 3 below illustrates the general dictionary structure, where the "outDict" can be seen in the browser "variable explorer", and with a double click, the other dictionaries can be accessed.

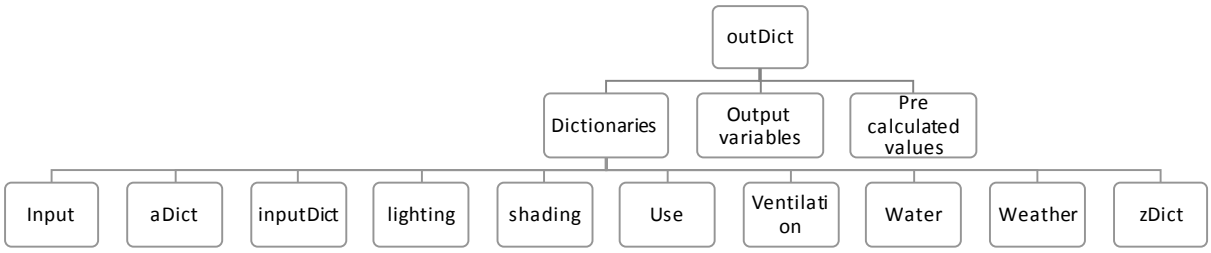

Figure 3: *General dictionary structure*

This general dictionary structure is followed for each step-by-step renovation measure, beginning from the *actual building status quo* until the *step n*. Figure 4 below illustrates the step-by-step dictionary structure, where the "outTable" can be seen in the browser "variable explorer", and with a double click, the other dictionaries can be accessed.

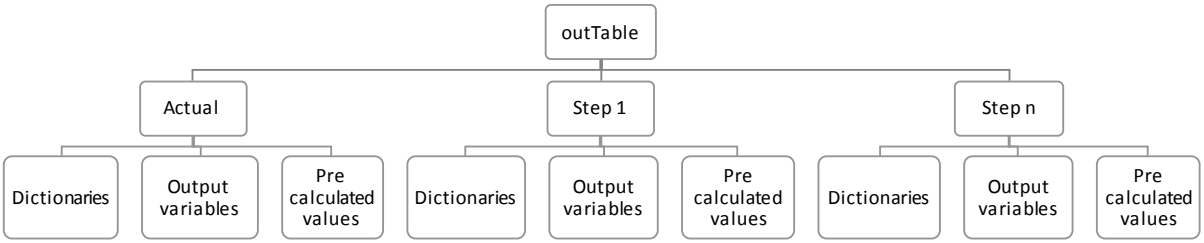

Figure 4: *Step-by-step dictionary structure*

Details regarding variable name, type and size in the dictionaries can be seen directly in the code.

Finally, the third section of the file **<accumulate_zones>** defines the output variables, which can also be passed into a JSON file. The current outputs are the same as required in the *iBRoad Roadmap Assistant* tool: energy needs for heating and cooling, final and primary energy demand and CO_2 emissions for lighting, heating, cooling, ventilation and domestic hot water, however this can also be adjusted by the code user.

The output parameters (calculation results) are, then, automatically exported to the file **<building_output.xlsx>.**

ii. How to run the code

Step 1

The first action is to enter the input parameter values in the file **<building_input.xlsx>**, sheet "Inputs" and column F "actual building status variable value". To help with the data entry, some comments and clarifications are included in column G "comments". If the user wants to calculate the energy demand of different variants in a step by step plan, the input parameters should be entered from column H "step 1 variable value" onwards, depending on how many steps are necessary. For example, an energy auditor wants to calculate a 2-step renovation roadmap: first step, adding roof insulation, and second step, replacing the windows to more energy efficient ones. For this, he/she enters in the column F all input data regarding the actual building status. In column H, he/she enters the new values related to step 1 (for example, U-values for the new roof), and in column I he/she enters the new values related to step 2 (for example, U- and g-values of the new windows). After entering all the values, the user should close and save this file.

Step 2

The next action is to specify in the file **<main.py>**, first part "user input", which energy demand calculation procedures should be applied: one step (only actual building statues energy calculation) or step by step approach (with more than one step). For that, the user enters "1" in code line 7 (b_actual =1) to calculate the energy demand using the actual building information from the <building_input.xlsx> file. For step-by-step calculation, the user has to specify in line 9, how many steps should be calculated. For example, for a two steps renovation plan (b_steps=2).

If the files are in an external folder, the user can specify in line 15 the folder's path variable beginning with r: path= r"path_name", but if no path needs to be specified, then path=0 (the code runs with both possibilities).

When the calculation is ready, a message **"All Done!"** is shown in the IPython console, and the results for the output parameters useful energy for heating and cooling, final and primary energy as well as CO_2 emissions for heating, cooling, domestic hot water, lighting and ventilation are shown in the file **<building_output.xlsx>**.

Step 3 (optional)

After the first run, the use can control and overwrite some indirect inputs in the file <building_input.xlsx>, worksheet "Rewritable Inputs".

Below some examples of the rewritable parameters:

- internal gains for person

- internal gains work activity
- room set temperature cooling
- room set temperature heating
- daily operating hours for heating
- height of room use level from the norm
- yearly operation days of heat, cooling and ventilation system
- yearly use days of the zone
- minimum outside air flow

Going back to step 1, after entering all the values, the user should close and save this file, and run the code in the file <main.py> again.

Step 4

Open the file <building_output.xlsx> to view the results.

iii. Where the code is hosted

The permanent repository for the code presented here may be found at
https://eeg.tuwien.ac.at/gitlab/iBROAD/iBROAD

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 754045