# Generic architecture of the framework

Concepts:

* The framework shall be flexible for different evaluation and calculation tasks, therefore the language must be selected accordingly.
* There must be an option to generate a full chain of tasks and as such ‘to freeze’ a given status of the developed scripts and functions. This ensures reproducibility.
* There must be options to reuse generic functions.
* A common architecture must be funded and followed by all later developers.
* A flexible framework shall be created which supports converting multiple files, multiple data formats, evaluation of the converted files with different evaluation profiles, and to do the two previous step at the same time.

Design decisions:

* The framework is python based. This means there is a possibility to solve all challenges in python. The framework cover layer is written also in python.
* There is an option to add MATLAB scripts and functions and integrate it to the python cover.
* The data will be handled as mat files within the process flow. For this a temporary folder is used within the framework.

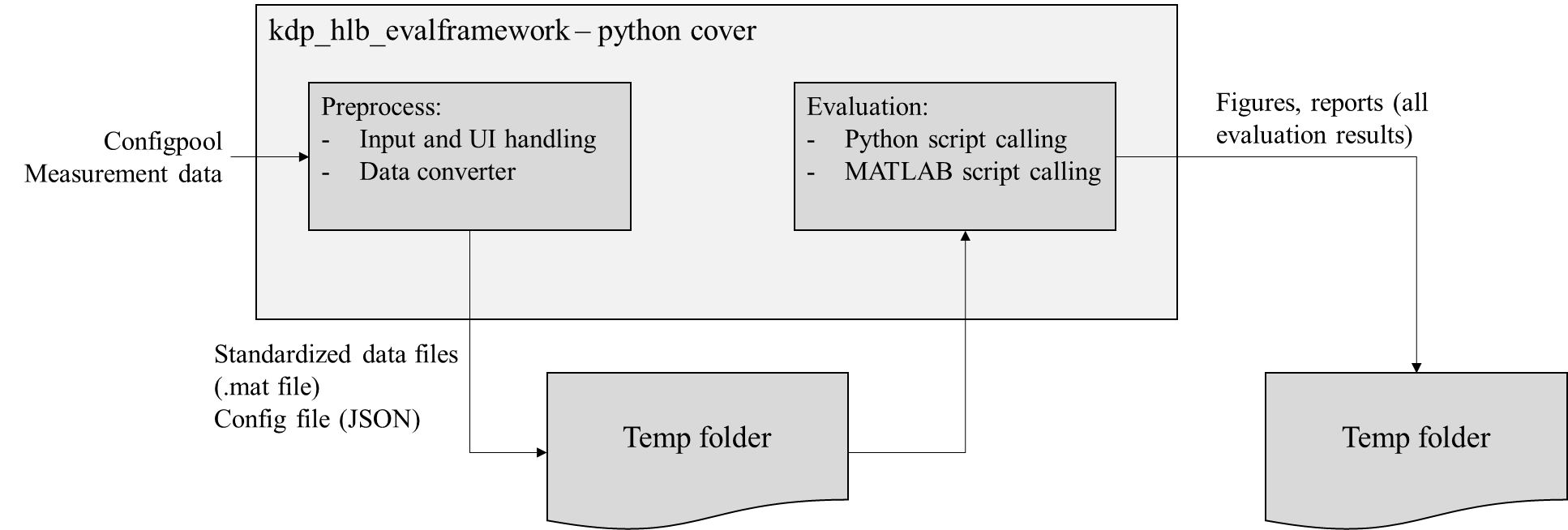


Figure : evaluation framework architecture

The interface files (config.json and measuremend mat files) are stored in the temp folder (within the repository). The outcome of the evaluation is stored in the temp folder as well, in forms of reports, figures, data files…etc. The architecture can be seen in Figure 1.

In the followings, the basic usage of the framework is introduced.

## Development usage

If you have tasks with intensively changing evaluation scripts, it is advised to use only part of the chain. For example, only MATLAB scripts are used with direct running, or python scripts running from command line. Even in this case, the internal evaluation architecture should be respected. The internal architecture can be seen in Figure 2.

For all configuration, 1 segmentation profile and 1 evaluation profile must be defined. One profile may contain subfunctions which are used by mutliple other profiles.

### Segmentation

Segmentation consists of a segmentation profile. This includes the file reading (mat file) and its preprocess. The following functions shall be implemented in here:

* Processing (e.g. filtering, invalid data cutting…etc)
* Labelling
* Quality checking and implausible data cutting
* New signal calculations, which are needed for the evaluation.

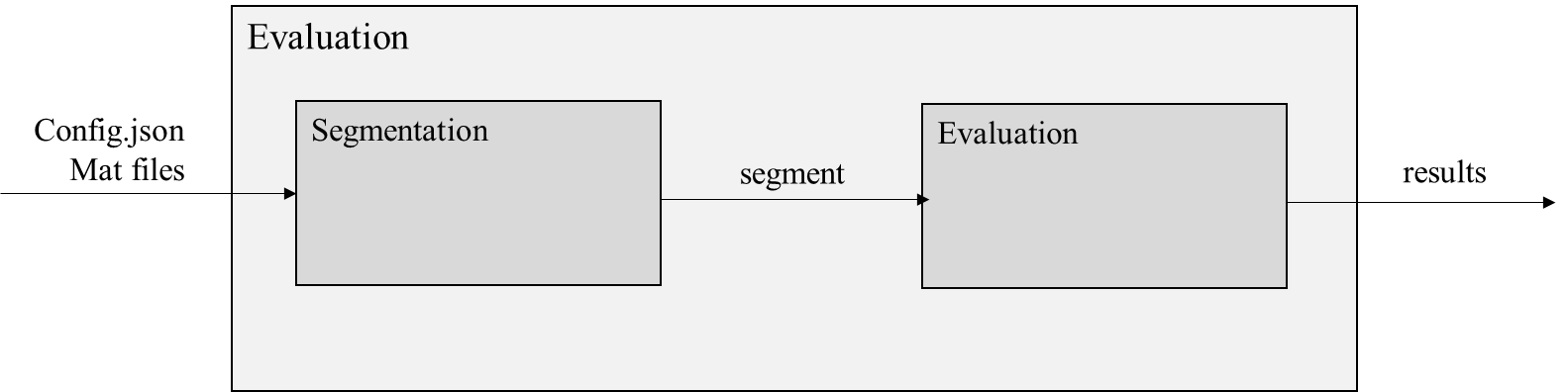


Figure : internal evaluation architecture

There can be an extra step (if needed) to save the segmented data back to mat files, and running another segmentation and evaluation profile combination for the process.

Examples of segmenting functions:

* Read mat file(s)
* Checking lane information validity
* Checking GPS signal validity
* Adding lane change channel to data

### Evaluation

Evaluation is also described by a profile. The profile will get the segment produced by the segmentor and has an output of either a set of figures, numerical files, reports…etc. These outputs shall be stored in the temp folder.

An evaluation profile can also contain calculations, like optimization, equations solving, or simulation options both open loop or closed loop. It must always be decided what the outcome is.

When there is a development ongoing, only the evaluation part of the chain can be run, but this still contains the segmentation and the evaluation scripts together.

## User level

If there is a python script, this must be integrated into the evaluation.py file. If there are MATLAB evaluation scripts, these should be compiled to a python package first, based on 2.1 attachment. Then, this should be integrated into evaluation.py file.

The chain must run through the GUI called by main.py. Here the right segmentation and evaluation profile must be selected.

Note: currently the whole chain running is still under development.

# Prerequisities

The following tools are needed for usage / development:

* MATLAB 2019b 64 bit
* Python 3.7
* Pycharm 2021.3
* Clone kdp\_hlb\_evalframework repository: <https://sourcecode01.de.bosch.com/projects/VMCBPTEAM/repos/kdp_hlb_evalframework/browse>
  + Develop branch shall be used

# Setting up the python based framework

This is especially needed for the whole evaluation chain running or the measurement converter. Clone the kdp\_hlb\_evalframework repository. Install the following python packages:

* Everything in \_converter/requirements.txt
* Matlab packages stored in \_matlab\_compiled\_packages

Once the installation is done, run the main.py without any parameters. The GUI should open up. This is seen on Figure 3. Measurement folder shall be any external folder with the .mf4, .mat or .bag (rosbag) files. The vehicle config will determine how the measurement signals are converted (this contains the list of the signals available in the given vehicle profile). Segmentation and evaluation profiles will be contained by the config file and this will be handed over to the evaluation scripts. Once everything is set up, click start, and the chain will run. The status will be reflected in a pop-up window.

Note: delete temp files are currently ineffective, all temporary files are stored and temp folder. After each rerun, the converted files and the evaluation results are deleted and regenerated.

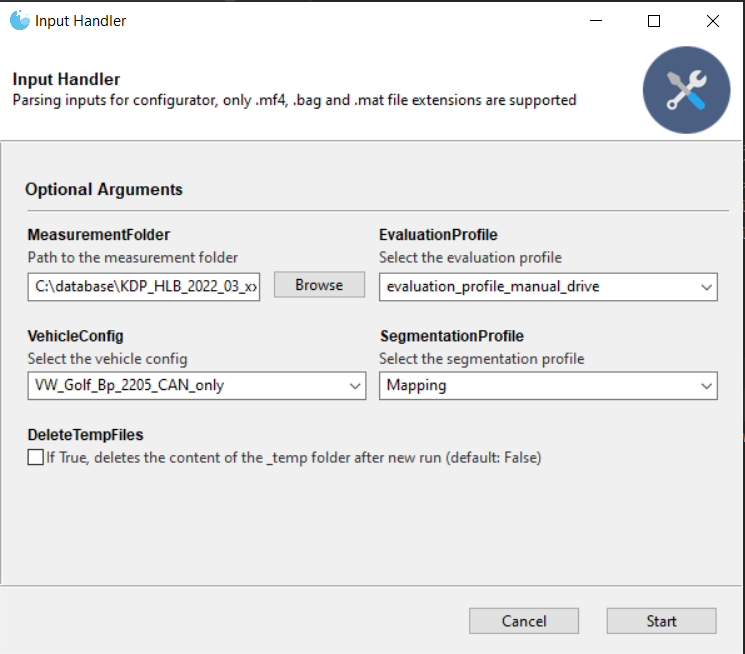


Figure 3: evaluation framework GUI

## Adding python based evaluation

Evaluation or calculation files written in python can be added under \_python\_evaluation folder. Subfolders shall be used. The segmentation and evaluation scripts must be called from the python\_evaluation.py file. The python\_evaluation receives the input from the preprocess: the configuration. The python\_evaluation first reads all available data from the temp folder. This will be stored in an internal variable. The the config and the data can be passed to the segmentor and the evaluator. These segmentor and evalautor functions are self written.

If needed, only the python\_evaluation.py can be called with the config parameter alone (without GUI, converter or any preprocess). This will make development quicker. As the last step of all development the scripts must be integrated into the full chain, though.

## Adding MATLAB based evaluation

The matlab\_evaluation\_cover.m file connects all segmentors and evaluators. First the config is read, then the segmentor function is called, and then the evaluator. The segmentor and the evaluator contains a switch case logic, which reads the profile configurations and call the appropriate segmentor and evaluator.

The segmentor contains the mat file read and returns the segment struct. The members of the segments struct are usually further structs containing the time and data channels. The segments struct is handed over the evaluator, which will save the results into the temp folder. The developer shall add new segmentor and/or evaluator profiles and add a case into the segmentor.m and the evaluator.m functions.

There are library items (prepared MATLAB functions) and separate segmentor and evaluator subfunctions which can be reused for your own develoment. It must always be repsected that these functions may be used by other existing segmentors or evaluators.

After adding the MATLAB functions, the matlab\_evaluation\_cover.m file can be called on its own for development purposes. For delivery purposes please follow attachment 4.1 to compile the matlab package. Then, the cmplete chain from the main.py can be called.

# Attachments

## Documentation/matlab2python\_package.pdf guideline – how to generate a python package from the MATLAB function.

**Document History**

* Note: The section document history is marked as “HIDDEN text”; hence it will not be printed: For this, you have to click the check box “Hidden Text” under “Tools -> Options”, in the tab “Print”.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Document Version | Compo-nent Version | Date | Author | Description | Status |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| Keywords |  |
| Roles |  |
| Screens |  |
| Process steps |  |
| Document type |  |