Course Unit Software Language Engineering

Masters in Informatics and Computing Engineering (M.EIC)

Department of Informatics Engineering

University of Porto/FEUP

1º Semester – 2024/2025

A DSL for Data Extraction from Arbitrary Sources

Assignment #2

Bob is working on a project that uses several tools and produces different output files each time he runs an experiment. As a starting point, he would like to consolidate the results of the different files on a table with a single row (besides the header), and output the table to an HTML file, for easier visualization.

He configured his tool-flow so that for each run of the experiment, the report files are copied to a new folder. This folder contains three report files, "vitis-report.xml", "decision_tree.yaml" and "profiling.json":

- *"vitis-report.xml": needs to extract the table under the element <Resources>, that is under the element <AreaEstimates> (there is another element <Resources>, under the element <Device>);*
- *"decision_tree.yaml": needs to extract the key-value pairs at the root level whose values are not composite types (example of composite type: object, array), as well as the key-values of the object "params";*
- *"profiling.json": needs to extract the name of the function that took a higher percentage of execution time, as well as the corresponding percentage;*

After extracting the values from the reports, they should be concatenated into a table with a single row (excluding header), with a new column at the beginning with the name of the folder that contains the report files. This table is then written to an HTML file.

By the end of this assignment, it is expected that you have:

- An expanded version of the semantic model which includes support for more file types, both input and output.
- An internal DSL that allows to configure and execute the semantic model.

Tips:

- Since you already have a configuration file that configures and executes the semantic model, you might choose to continue supporting the configuration file. If you can translate the configuration file to internal DSL commands, in the final assignment you can translate the external DSL to a configuration file, instead of building an interpreter for the DSL intermediate representation.

- If the configuration file is limiting the design of your API, you are not required to support a configuration file at the end of the assignment. In that case, for this assignment you can demonstrate your implementation by using Java code that calls your API.
- As you did not have to write a YAML parser, you also do not need to write a JSON parser or a XML parser to extract information from the JSON and XML file, use already existing classes/libraries (e.g. Gson, XStream).