

Francesca L. Bleken (SINTEF)

Louis Ponet (EPFL)

Casper W. Andersen (SINTEF)

Jesper Friis (SINTEF)

TRANSLATION AND OPENMODEL OIP: SEMANTIC MANAGEMENT OF MODELLING WORKFLOWS



© OPENMODEL CONSORTIUM
CONFIDENTIAL

EXECFLOW —

<https://github.com/H2020-OpenModel/ExecFlow>

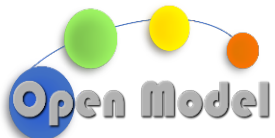
TOWARDS A SEMANTIC DESCRIPTION OF WORKFLOWS

Workflow executor that combines...

- use of pipelines (i.e. documented datasinks and sources) for data handling
- diverse ways of handling simulations
 - AiiDA/ExecFlow plugins designed for each sw
 - Has generic plugin for running executables
 - ++



... and provides full provenance.



Based on AiiDA



Other used technologies



OTEAPI – data documentation

<https://github.com/EMMC-ASBL/oteapi-core/>

DLite – interoperability framework

<https://github.com/SINTEF/dlite/>

WHAT WE HAVE SEEN ALREADY

Documentation of data with **partial pipelines**

Data sources

How to retrieve data

How to parse data into a datamodel
(our interoperability framework)

How to map data to a common language
(ontology)

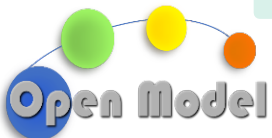
Data sinks

How to save/store data

How to generate data from a data
model in our interoperability framework

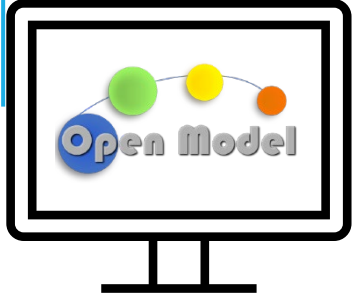
How to map data to a common
language (ontology)

How to combine partial pipelines to **pipelines** create desired input/output.



WHAT IS OUR USE CASE?

Q: What is the density of water?



Translates to make an instance of this

```
{
  "uri": "http://openmodel.eu/meta/0.1/WaterDensity",
  "description": "Entity describing water density",
  "dimensions": [],
  "properties": {
    "density": {
      "type": "float64",
      "description": "Water density"
    }
  }
}
```

Data Model

What is the density of water?

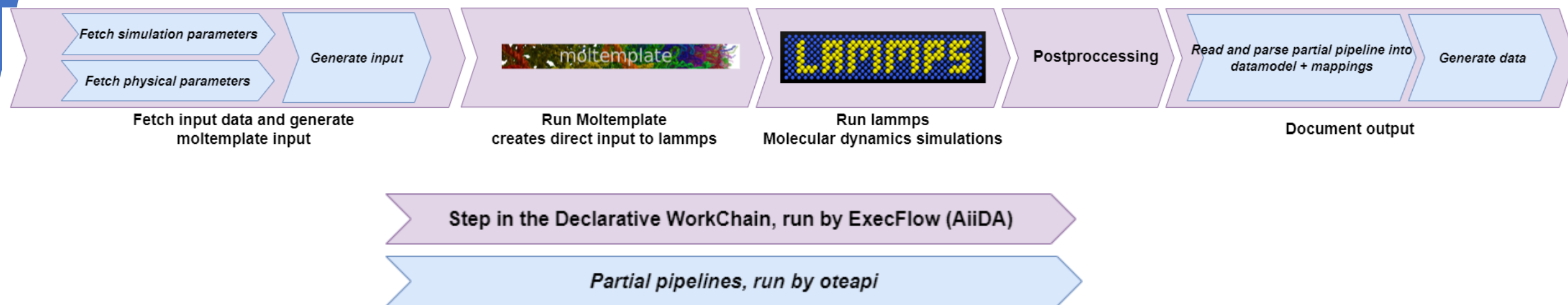
I don't know how.

We can run a simulation to find out.

OK, we can have OpenModel help us running simulations, let me show you

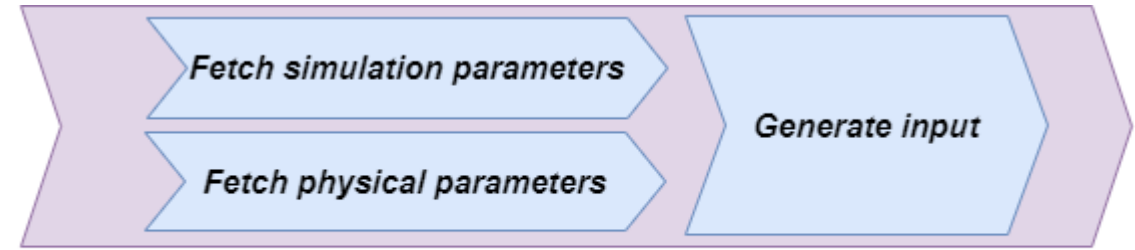
OntoFlow (built on OntoKB) will provide the workflow

WORKFLOW: DECLARATIVE WORKCHAIN



<https://github.com/H2020-OpenModel/Public/tree/OntoTrans2ndWorkshop/OntoTrans2ndWorkshop>

PIPELINES RECAP.



Fetch input data and generate moltemplate input

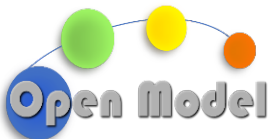
| | A | B | C | D | E | F | G |
|---|-----------------|---------------------|----|------------------|----------|----------|-------------|
| 1 | number_of_steps | equilibration_steps | s_ | production_steps | timestep | gridsize | gridspacing |
| 2 | 50 | 100 | 5 | 50000 | 10 | 4 | 6.4 |
| 3 | 20 | 200 | 5 | 500000 | 2 | 4 | 6.4 |
| 4 | 10 | 10 | 5 | 50 | 1 | 4 | 6.4 |

| | A | B |
|---|-------------|----------|
| 1 | temperature | pressure |
| 2 | 298.15 | 1000 |
| 3 | 200 | 800 |
| 4 | 150 | 600 |

Fetch simulation parameters

Fetch physical parameters

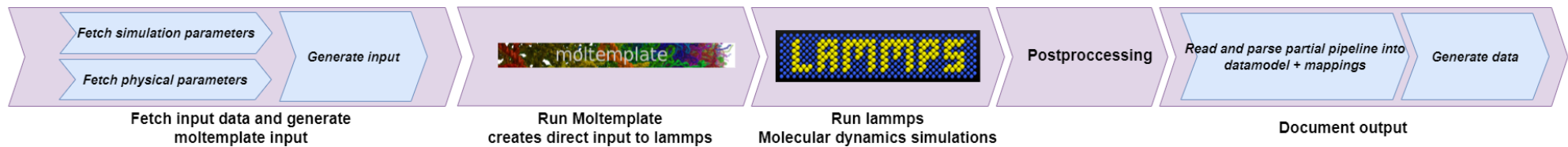
The datasource partial pipelines populate a «library» of DataModels and mappings



moltemplate

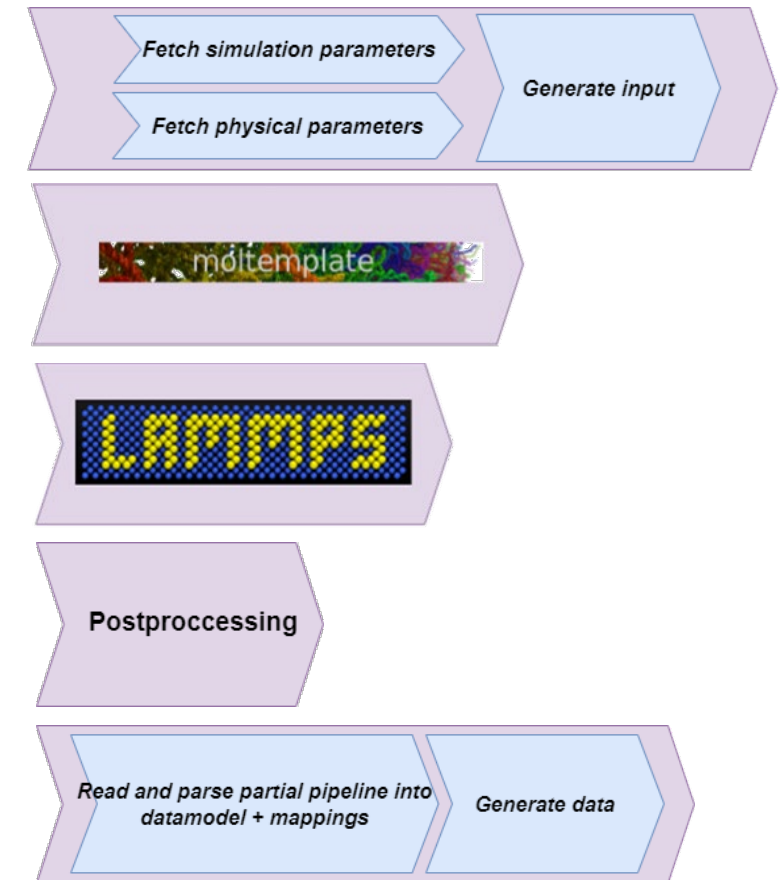
Generate input

The datasink pipeline has access to everything in the «library».



steps:

- workflow: `execflow.oteapipeline`
...
- workflow: `execflow.exec_wrapper`
...
- workflow: `execflow.exec_wrapper`
...
- calcjob: `execflowdemo.lammps.density`
...
- workflow: `execflow.oteapipeline`
...



RUNNING THE PIPELINE

Fetch simulation parameters

Fetch physical parameters

Generate input

```
- workflow: execflow.oteapipeline
inputs:
  pipeline:
    $ref: file:pipeline_input.yml
  run_pipeline: pipe
  from_cuds:
    - step01_input_file

postprocess:
  - "{{ ctx.current.outputs['collection_id']|to_results('collection_uuid') }}"
  - "{{ ctx.current.outputs.results['step01_input_file']|to_ctx('step01_input_file') }}" # This gives the PK number
```

Declarative WorkChain

Pipeline file name

Pipeline name

What to get back from the pipeline

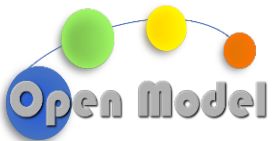
```
options: "mode=w"

- function: file2collection
  functionType: aiidacuds/file2collection
  configuration:
    path: /tmp/ExecFlowDemo/meso_multi_sim_demo/case_aiida_shell/output/step01_input2.lt
    label: step01_input_file

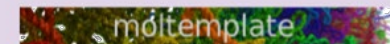
- function: cuds2datanode
  functionType: aiidacuds/cuds2datanode
  configuration:
    names: from_cuds

pipelines:
  pipe: load_data | mappings | load_data_1 | mappings_1 | mappings_3 | generate_moltemplate_input | file2collection | cuds2datanode
```

Pipeline



RUNNING EXECUTABLES



Declarative WorkChain

```
- workflow: execflow.exec_wrapper
inputs:
  command: "/tmp/ExecFlowDemo/meso_multi_sim_demo/case_aiida_shell/mock_executables/mockmoltemplate.sh"
  arguments:
    - "{infile1}"
    - "{infile2}"
    - "{infile3}"
  files:
    infile1:
      filename: "01_input_variables.txt"
      node: "{ ctx.step01_input_file }"
    infile2:
      filename: "tip4p2005_cg_01.lt"
      template: "/tmp/ExecFlowDemo/demo_pm/molc/tip4p2005_cg_01.lt"
    infile3:
      filename: "solvent_cg_creation.lt"
      template: "/tmp/ExecFlowDemo/demo_pm/molc/solvent_cg_creation.lt"
  outputs:
    - output.in
    - output.data
    - output.in.settings
    - output.in.run
```

Annotations:

- Path to executable (points to `command`)
- Arguments, point to the files below (points to `arguments`)
- Input files (points to `infile1`, `infile2`, and `infile3`)
- Output files (points to `output.in`, `output.data`, `output.in.settings`, and `output.in.run`)

RUNNING EXECUTABLES

LAMMPS

```
- workflow: execflow.exec_wrapper
  inputs:
    command: "/tmp/ExecFlowDemo/meso_multi_sim_demo/case_aiida_shell/mock_executables/mocklammeps.sh"
    arguments:
      - "-in"
      - "{in}"
    files:
      in:
        filename: "output.in"
        node: "{{ ctx.output_in }}"
      data:
        filename: "output.data"
        node: "{{ ctx.output_data }}"
      settings:
        filename: "output.in.settings"
        node: "{{ ctx.output_in_settings }}"
      run:
        filename: "output.in.run"
        node: "{{ ctx.output_in_run }}"
      init:
        filename: "output.in.init"
        node: "{{ ctx.output_in_init }}"
    outputs:
      - log.lammeps
```

Path to executable

Arguments, point to the files below

Declarative WorkChain

Input files

Output files

How are datanodes (AiiDA data) passed from one step to the other?



PASSING INFO & AIIDA CALCJOB

LAMMPS

Postprocessing

Declarative WorkChain

```
filename: "output.in.run"
node: "{{ ctx.output_in_run }}"
init:
  filename: "output.in.init"
  node: "{{ ctx.output_in_init }}"
outputs:
  - log.lammps
  - output.log
  - output.dump
```

End of specification of inputfiles to lammps

Outputfiles, that are cast to AiiDA DataNodes

```
postprocess:
  - "{{ ctx.current.outputs['log_lammps']|to_ctx('log_lammps') }}"
  - "{{ ctx.current.outputs['output_log']|to_ctx('output_log') }}"
  - "{{ ctx.current.outputs['output_dump']|to_ctx('output_dump') }}"
```

The DataNodes desired in a later step are stored in the **context**

```
- calcjob: execflowdemo.lammps.density
inputs:
  log: "{{ ctx.output_log }}"
postprocess:
  - "{{ ctx.current.outputs['density'] | to_results('density') }}"
  - "{{ ctx.current.outputs['density'] | to_ctx('density') }}"
```

Postprocessing job, own AiiDA calcjob

Get the desired input from the context

AIIDA CALCJOB & FINAL PIPELINE

Postprocessing

Read and parse partial pipeline into
datamodel + mappings

Generate data

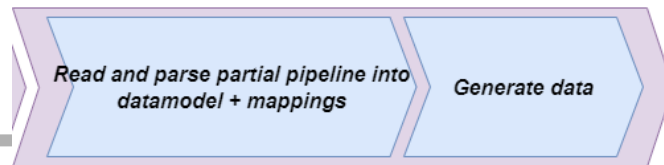
Document output

```
- calcjob: execflowdemo.lammps.density
  inputs:
    log: "{{ ctx.output_log }}"
  postprocess:
    - "{{ ctx.current.outputs['density'] | to_results('density') }}"
    - "{{ ctx.current.outputs['density'] | to_ctx('density') }}"
```

Declarative WorkChain

```
- workflow: execflow.oteapipeline
  inputs:
    pipeline:
      $ref: file:pipeline_output.yml
    run_pipeline: pipe
    to_cuds:
      - density
    density:
      "{{ ctx.density }}"
  postprocess:
    - "{{ ctx.current.outputs['collection_id']|to_ctx('collection_uuid') }}"
```

OUTPUT PIPELINE



Document output

```
- function: datanode2cuds
  functionType: aiiidacuds/datanode2cuds
  configuration:
    names: to_cuds
```

Pipeline

Pipeline

IN THE WORKFLOW:

```
- mapping: mappings
  mappingType: mappings
  prefixes:
    float: onto-ns.com/meta/1.0/core.float#
    demonto: http://openmodel.eu/meta/0.1/WaterDensity#
    emmo: http://emmo.info/emmo#
    map: http://emmo.info/domain-mappings#
  triples:
    - ["float:value", "map:mapsTo", "emmo:WaterDensity"]
    - ["demonto:density", "map:mapsTo", "emmo:WaterDensity"]
```

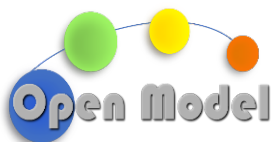
```
- function: dump_output
  functionType: application/vnd.dlite-generate
  configuration:
    datamodel: http://openmodel.eu/meta/0.1/WaterDensity
    driver: yaml
    location: /tmp/ExecFlowDemo/meso_multi_sim_demo/case_aiida_shell/output/test.yaml
    options: "mode=w"
```

pipelines:

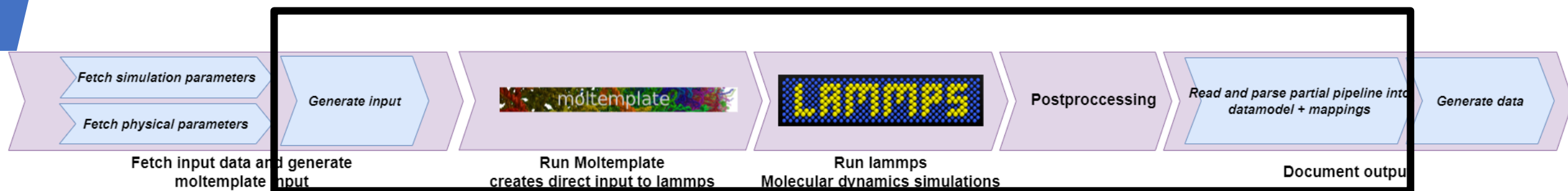
```
pipe: datanode2cuds | mappings | dump_output
```

Verdi process list -all

Verdi process node show/attributes PKnumber



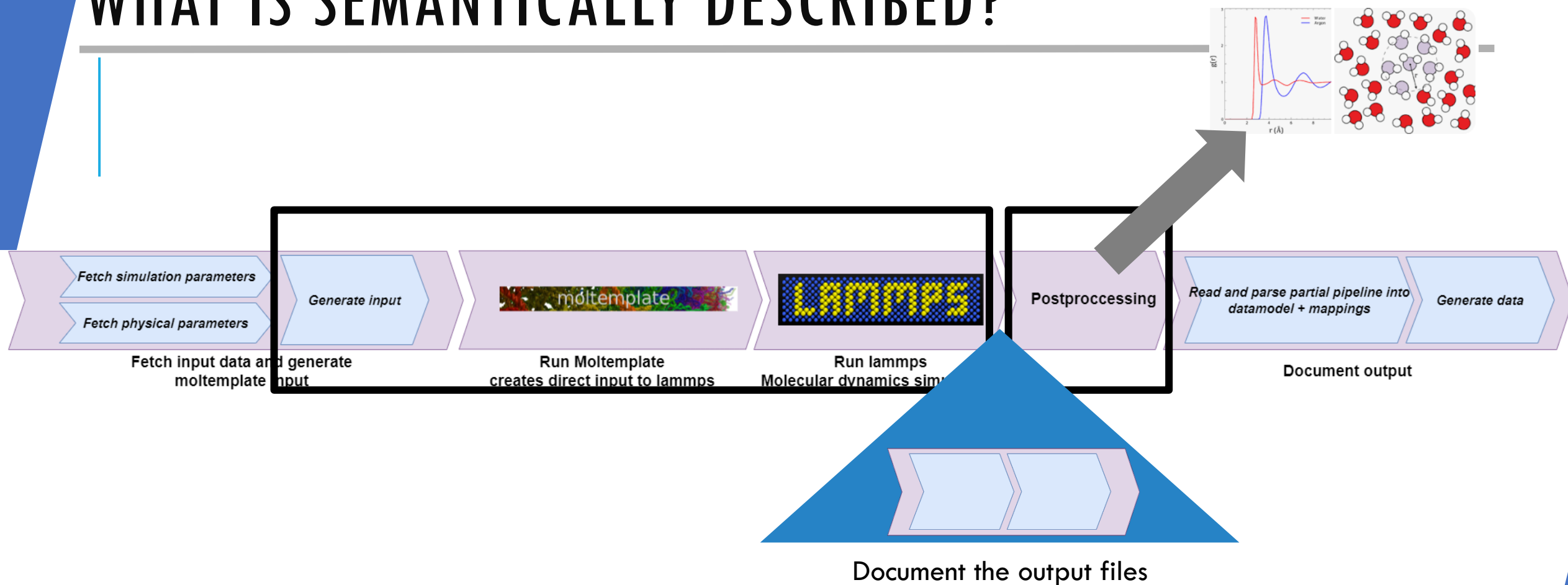
WHAT IS SEMANTICALLY DESCRIBED?



Step in the Declarative WorkChain, run by ExecFlow (AiiDA)

Partial pipelines, run by oteapi

WHAT IS SEMANTICALLY DESCRIBED?



→ Now free to match the first part with a different analysis than density...

Step in the Declarative WorkChain, run by ExecFlow (AiiDA)

Partial pipelines, run by oteapi

OntoTrans 2nd –Workshop 07.08.2023


RUN THE EXAMPLE

```
"uri": "http://openmodel.eu/meta/0.1/WaterDensity",  
"description": "Entity describing water density",  
"dimensions": [],  
"properties": {  
    "density": {  
        "type": "float64",  
        "description": "Water density"
```

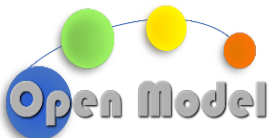
Data Model

```
datamodel: http://openmodel.eu/meta/0.1/WaterDensity  
driver: yaml  
location: /tmp/ExecFlowDemo/meso_multi_sim_demo/case_aiida_shell/output/test.yaml  
options: "mode=w"
```

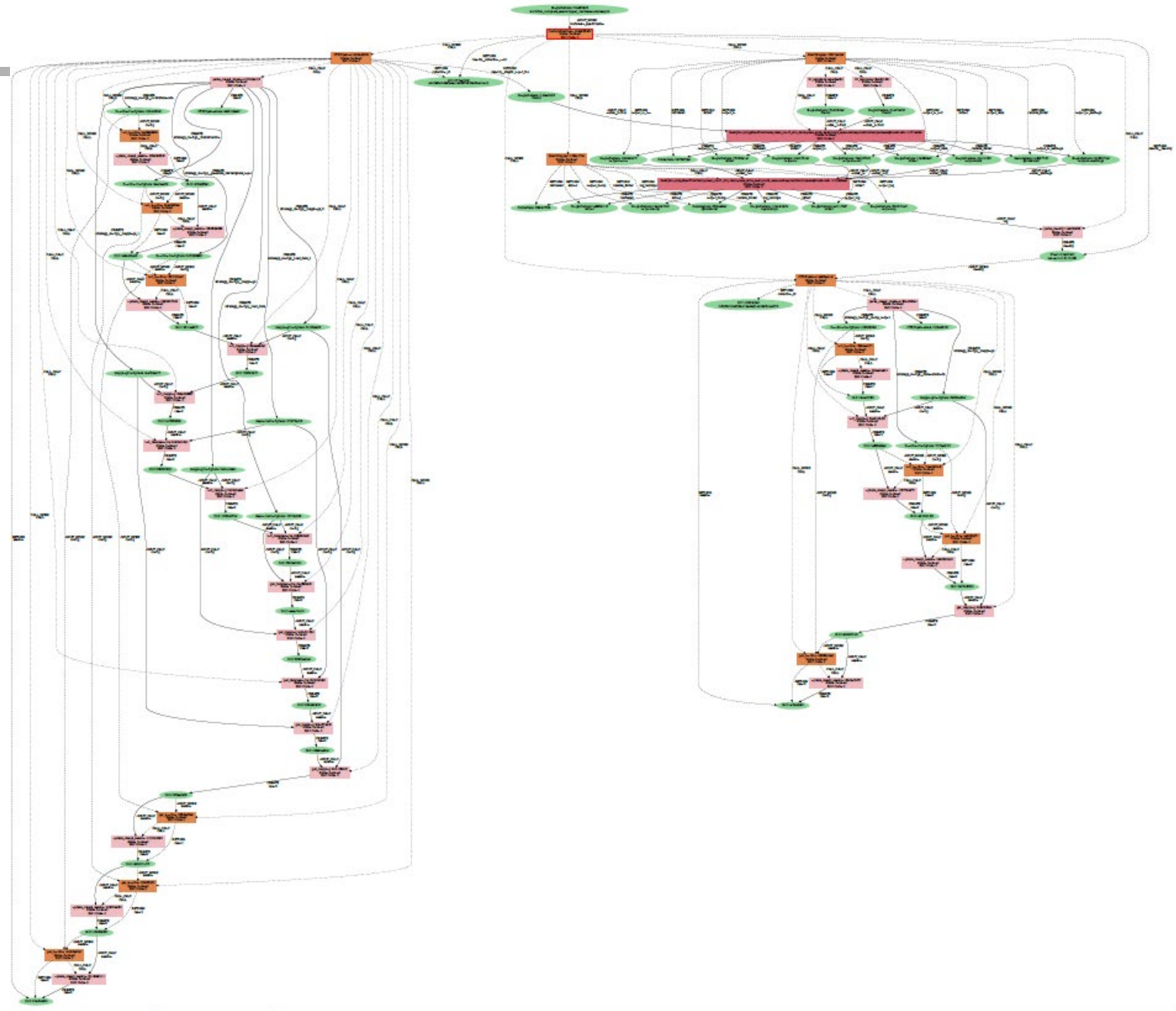
Pipeline



```
e2bce8a5-1337-4520-9037-70321383ce23:  
  meta: http://openmodel.eu/meta/0.1/WaterDensity  
  dimensions: {}  
  properties:  
    density: 1.0442329363636  
output/test.yaml (END)
```



Provenance graph





This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 953167.

This document and all information contained herein is the sole property of the OpenModel Consortium. It may contain information subject to intellectual property rights. No intellectual property rights are granted by the delivery of this document or the disclosure of its content. Reproduction or circulation of this document to any third party is prohibited without the consent of the author(s).

The content of this deliverable does not reflect the official opinion of the European Union. Responsibility for the information and views expressed herein lies entirely with the author(s).

All rights reserved.