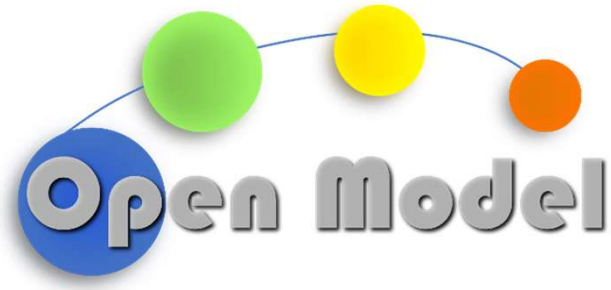
Otello M Roscioni (GCL)
Gerhard Goldbeck (GCL)
Arrigo Calzolari (CNR-NANO)



Encompassing materials modelling for industry 4.0

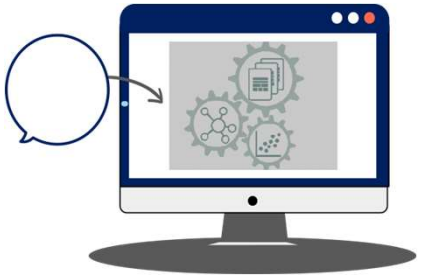
24 September 2024

The OpenModel project

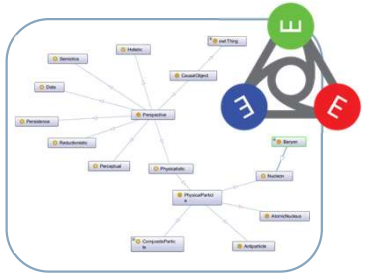
"Where have I seen this before?"

OpenModel is a **management system** for materials modelling workflows based on **semantic** technologies:

- Interoperability between physics-based models, solvers, third-party software, post-processors, and databases via a semantic layer.
- Automatic generation and execution of materials modelling workflows.
- FAIR data creation.



User front-end



Ontologies

AiiDA

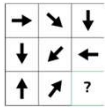
SimPhoNy

DLite

Workflows and data management

DB

Software



Reasoning

V&V Services

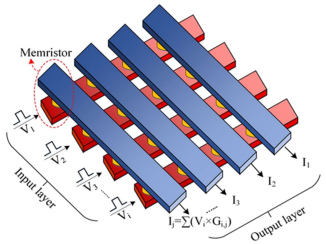
Third-party tools

Motivation and Background

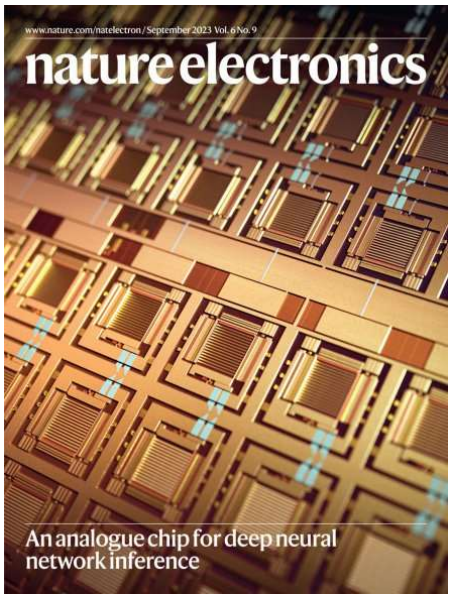
- **Materials modelling** enables the prediction of material properties, allowing for the **design of new materials** with tailored properties, which is critical in Industry 4.0 where customization and innovation are key.
- The data generated by materials modelling can be used to inform **data-driven business decisions**, making it a fundamental component of Industry 4.0's digital economy.

Synaptic electronics

- Overcoming the limitations of Si-based CMOS electronics architecture: power consumption and memory wall.
- Artificial neurons and synapses are considered essential for the progress of future brain-inspired computing, outperforming von Neumann architectures.
- **Solution:** in-memory computing where data are processed within the memory element integrated in the back-end-of-line.
  - Memristor (PCM, ReRAM, FeFET, etc).



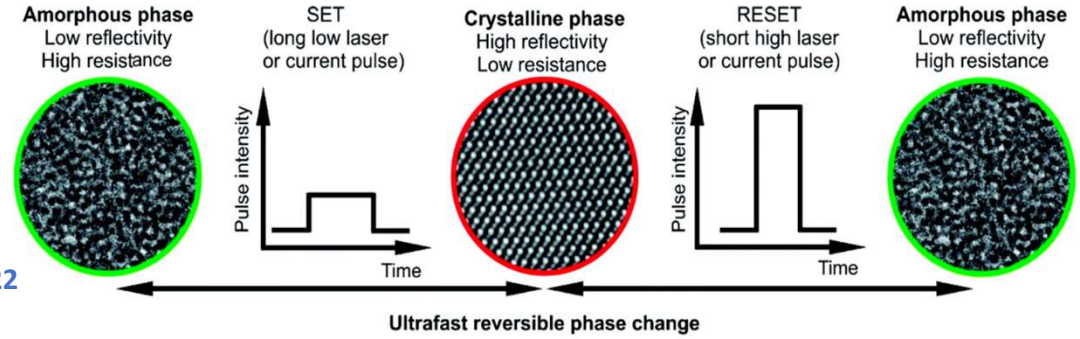
Md. Oli-Uz-Zaman et al.
J. Low Power Electron. Appl. **2022**



September 2023

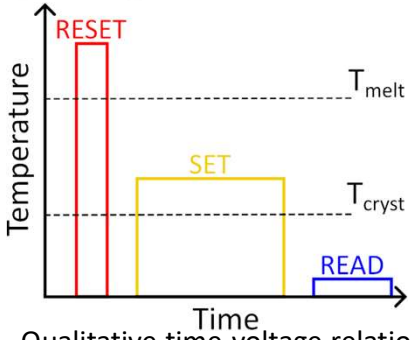
Phase-change materials (PCM)

Non-volatile memory devices retain their status when the power is off. They exploit the unique property of certain materials to switch between amorphous and crystalline states.

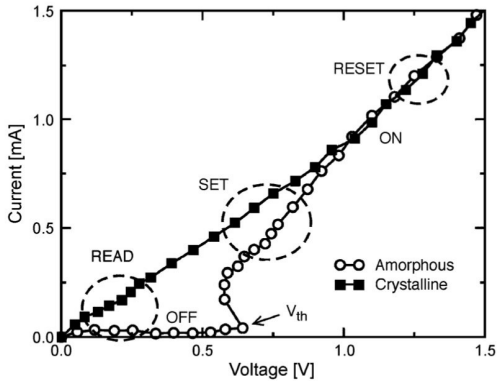


A. Ehrmann et al. *Appl. Res.* **2022**
DOI: 10.1002/appl.202200024

- Reference Phase Change Material: $Ge_2Sb_2Te_5$ (GST225).
- Crystallization temperature T_{cryst} = 520 K.



Qualitative time-voltage relation for the phase change.

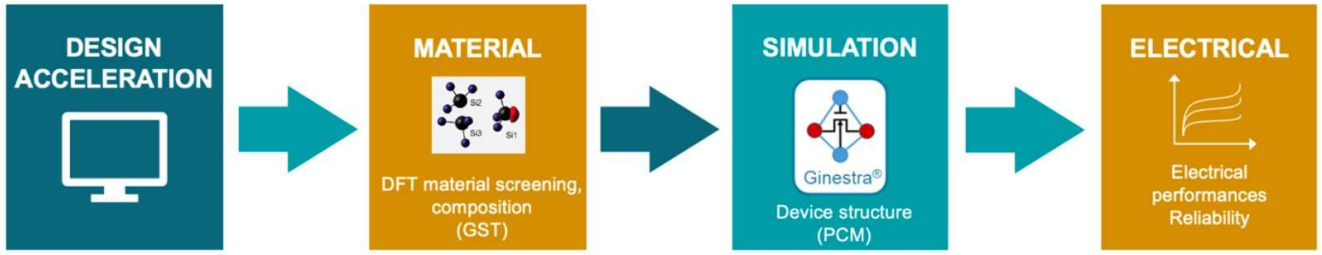


A use case for materials modelling

Problems

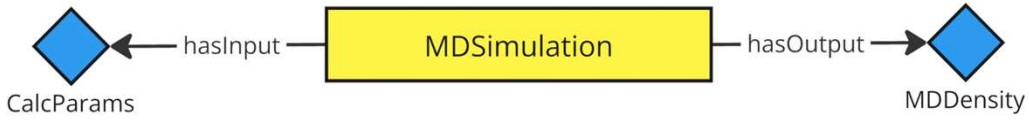
- Overcoming the limitations of Si-based CMOS electronics architecture: power consumption and memory wall.
- Artificial neurons and synapses are considered essential for the progress of future brain-inspired computing, outperforming von Neumann architectures.
- **Solution:** in-memory computing where data are processed within the memory element integrated in the back-end-of-line.
  - Memristor (PCM, ReRAM, FeFET, etc).
- **Challenges:**
  - Device geometry difficult to optimize.
  - Parameters for new materials not available.

The interplay between materials and their influence on the device is hard to determine. Need for **co-designing** materials and device architecture.

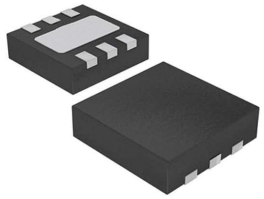


Semantic description of workflows

- The topology of a workflow is build using **hasInput** and **hasOutput** relations connecting processes to datasets.



- The input and output **datasets** are "decorated" with **semantic relations**, e.g.

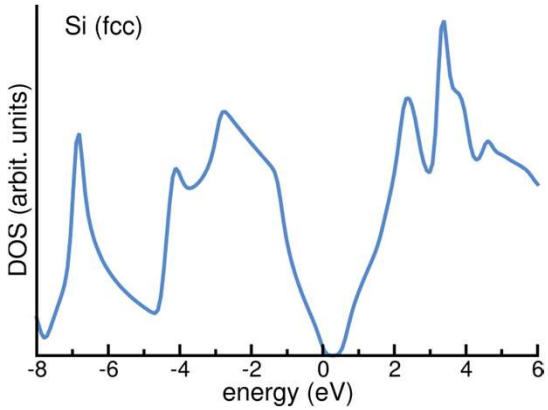


MOSFET

hasProperPart

Silicon

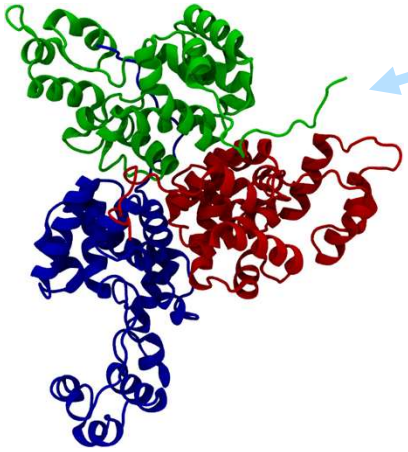
hasProperty



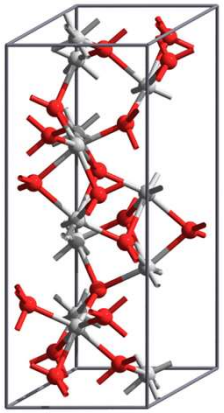
- Density of States
- Temperature
- Permittivity

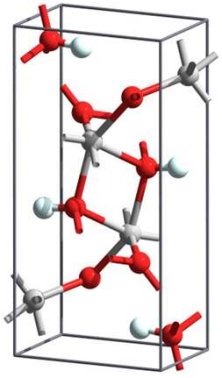
Breaking workflows into single tasks

- Each workflow is broken down into a series of individual tasks.
- Each task is a computational **process** transforming input data into output data.
- Each process has a **limited scope** and is described with a **wrapper**.
- **Same** software, **different uses**: `pw.x < pw.sample.scf.in > pw.sample.scf.out`

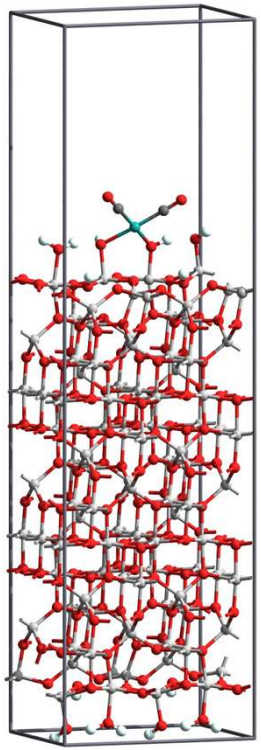


4OK7 protein, 5680 atoms





Crystals

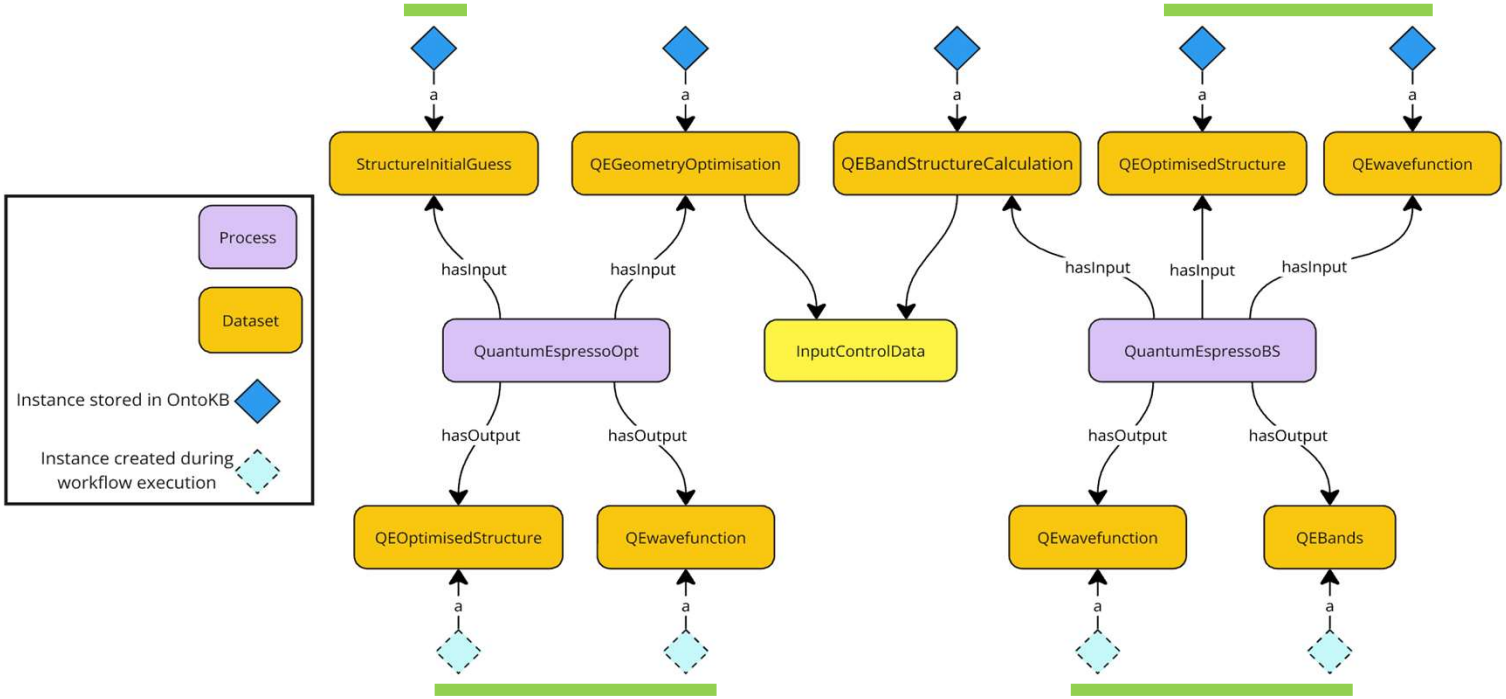


Surfaces

Different tasks:

- Geometrical optimisation
- Band-structure

Describing Quantum Espresso tasks



- Tasks are divided into different classes, that use **hasInput** and **hasOutput** relations to generate different topologies.
- How to describe the input datasets with semantic relations?

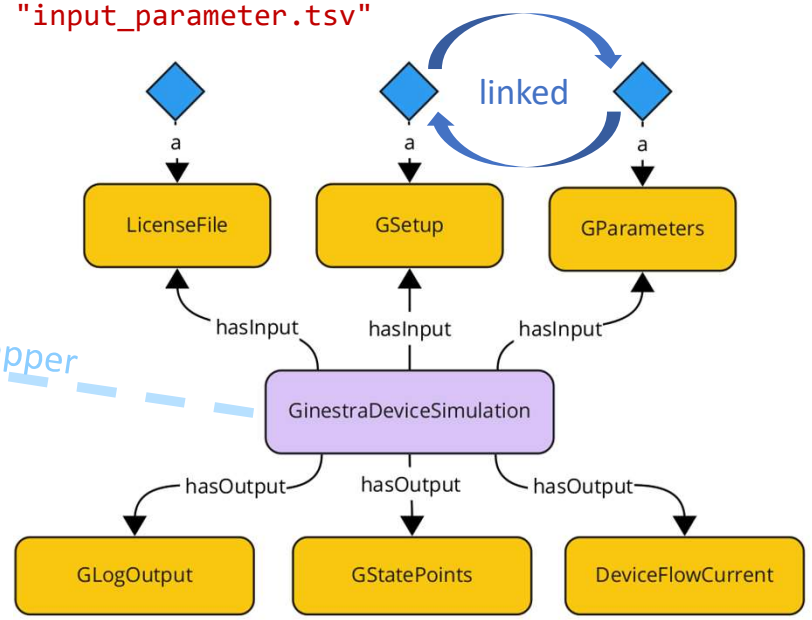
Describing the input datasets of a Ginestra task

The I-V curve of a transistor where the insulator is set, and the semiconductor can be changed.

Suppose we want to describe a simulation that is executed from the CLI:

```
$ ginestra-core-sim.exe -l "ginestra.lic" -i "input.ini" -p "input_parameter.tsv"
```

```
- workflow: execflow.exec_wrapper
    inputs:
      shelljob:
       metadata:
         options:
           resources:
             num_machines: "{{ ctx.ginestra.m_n_machines }}"
             num_mpiprocs_per_machine: "{{ ctx.ginestra.m_n_mpiprocs_pm }}"
      command: "{{ ctx.ginestra.executable }}"
      arguments:
        - "-l"
        - "{{ ctx.ginestra.licence }}"
        - "-i"
        - "{{ ctx.ginestra.g_setup }}"
        - "-p"
        - "{{ ctx.ginestra.g_parameters }}"
      outputs:
        - test.log
    postprocess:
      - "{{ ctx.current.outputs['test_log']|to_ctx('g_output') }}"
```

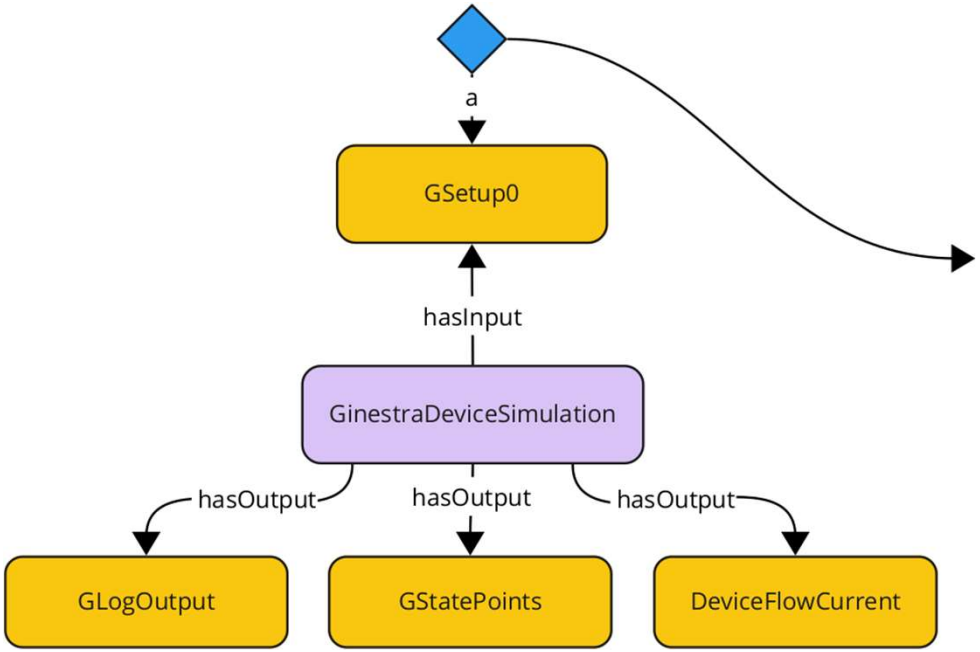


- Where are the semantics?

Hiding the content of the datasets

Suppose we are not interested in knowing the content of the input data.

Then the input can be stored in a **single dataset**, representing a particular instance of a *device* and its parameters.



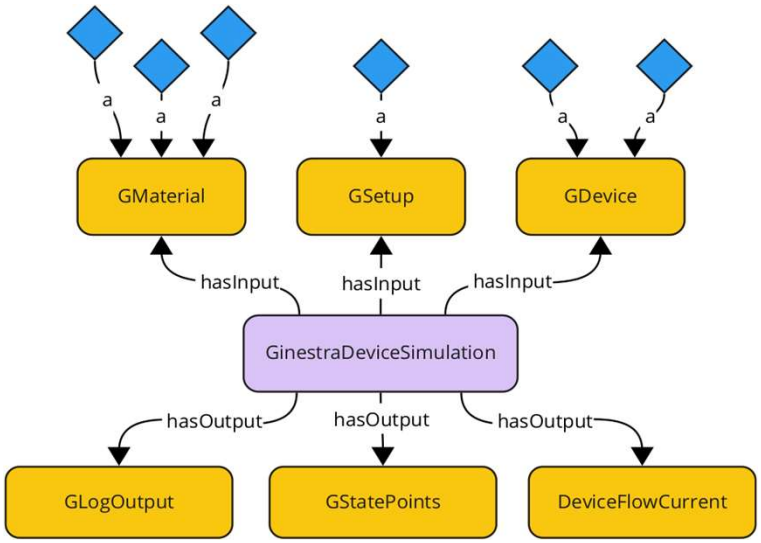
Input dataset

```
---
uFDE13MVbM:
  meta: http://ontotrans.eu/meta/0.1/InputGinestraSimulation
  dimensions: {}
  properties:
    executable: /tmp/ss1/dummy/ginestra-core-sim.exe
    licence: /tmp/ss1/Inputs/ginestra.lic
    g_input: /tmp/ss1/Inputs/input.ini
    g_parameters: /tmp/ss1/Inputs/input_parameter.tsv
    m_n_machines: 1
    m_n_mpiprocs_pm: 1
...
```

Revealing the content of the input datasets

The input data is **sorted** into a dataset referring to the active material, one referring to the device, and one with numerical parameters.

```yaml
---
nThlFwkFpI:
  meta: http://ontotrans.eu/meta/0.1/GinestraMaterial
  dimensions: {}
  properties:
     bandgap: 5.0
     effective_mass: 0.5
...
```



Input datasets

```yaml
---
Tf0GCu7eH3:
  meta: http://ontotrans.eu/meta/0.1/InputGinestraSimulation
  dimensions: {}
  properties:
     executable: /tmp/ss1/dummy/ginestra-core-sim.exe
     licence: /tmp/ss1/Inputs/ginestra.lic
     m_n_machines: 1
     m_n_mpiprocs_pm: 1
     input_template: /tmp/ss1/Inputs/ginestra_input.template # generic
     state: /tmp/ss1/Inputs/state.hdf5
     out_folder: ./
     output: sim_device.hdf5
     grid_size: 100
...
---
4I4DsZ4Kjj:
  meta: http://ontotrans.eu/meta/0.1/GinestraDevice
  dimensions: {}
  properties:
     g_device: /tmp/ss1/Inputs/device1.xml
     g_test: /tmp/ss1/Inputs/test1.xml
     param_template: /tmp/ss1/Inputs/ginestra_input_parameters1.template # test1
     temperature: 298.5
...
```

From file `input.ini`

Adding semantics with ontological data mapping

The semantic mapping now allows to create dataset classes for **materials** and **devices**, enabling classification and filtering of the information stored in the Knowledge Base.

Data mappings

```
mappings = [
  (GDEV,                EMMO.isDescriptionFor, EMMO.ManufacturedProduct),
  (GDEV,                EMMO.hasProperPart,    SS1.VirtualMaterial),
  (GDEV.g_device,       MAP.mapsTo,            SIMSOFT:XMLFile),
  (GDEV.g_device,       EMMO.isDescriptionFor, EMMO.ManufacturedProduct),
  (GDEV.g_test,         MAP.mapsTo,            SIMSOFT:XMLFile),
  (GDEV.param_template, MAP.mapsTo,            SIMSOFT:Jinja2File),
  (GDEV.temperature,    MAP.mapsTo,            EMMO.ThermodynamicTemperature),
  (GDEV.temperature,    EMMO.isDescriptionFor, EMMO.ManufacturedProduct)
]
```

```
mappings = [
  (GMAT,                EMMO.isDescriptionFor, SS1.VirtualMaterial),
  (GMAT.bandgap,        MAP.mapsTo,            SS1:BandGap),
  (GMAT.bandgap,        EMMO.isPropertyFor,    SS1:VirtualMaterial),
  (GMAT.effective_mass, MAP.mapsTo,            EMMO:EffectiveMass),
  (GMAT.effective_mass, EMMO.isPropertyFor,    SS1:VirtualMaterial)
]
```

Input datasets

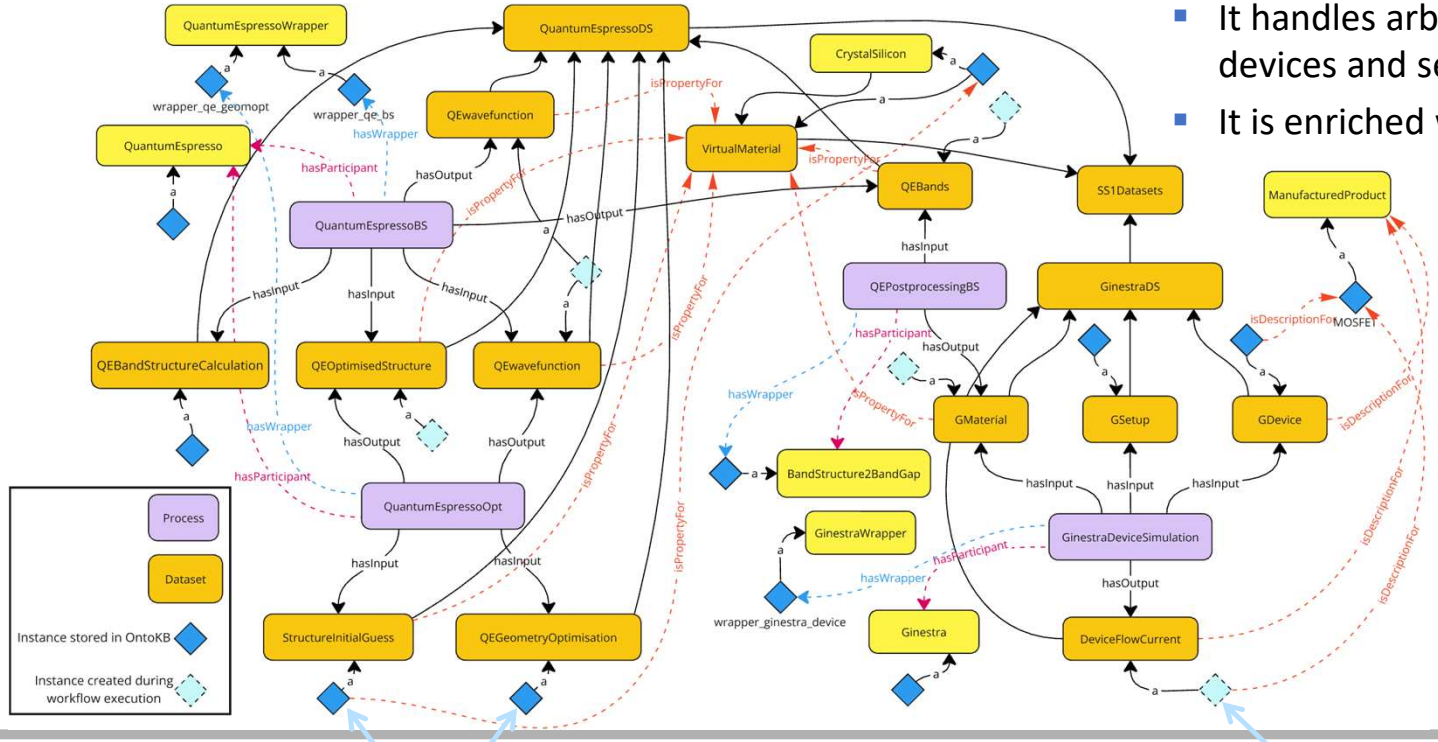
```
---
4I4DsZ4Kjj:
  meta: http://ontotrans.eu/meta/0.1/GinestraDevice
  dimensions: {}
  properties:
     g_device: /tmp/ss1/Inputs/device1.xml
     g_test: /tmp/ss1/Inputs/test1.xml
     param_template: /tmp/ss1/Inputs/ginestra_input_parameters1.template # test1
     temperature: 298.5
...
```

```
---
nThlFwkFpI:
  meta: http://ontotrans.eu/meta/0.1/GinestraMaterial
  dimensions: {}
  properties:
     bandgap: 5.0
     effective_mass: 0.5
...
```

- OTEAPI pipelines to serialise the input datasets into the correct files.

Managing workflows: the Knowledge Base

- It creates and stores FAIR data.
- It can be queried.
- It connects different software and documents the resulting workflows.
- It handles arbitrary combinations of devices and semiconductors.
- It is enriched with additional relations.



Conclusions

- A workflow for a **multiscale simulation** of device and materials.
- Data and process **documentation** with semantic technologies: ontologies and triplestores.
- To document workflows, begin by setting the **modelling goals**:
  - Set the level of detail for the input and output datasets.
  - Separate syntax from semantics.
  - Create the classes for processes and datasets,
  - … and then the wrappers and pipelines.
- Part of the **EMMO** ecosystem: https://emmo-repo.github.io/
- Finds out more at https://github.com/**H2020-OpenModel**/

Beware of semantics

actually, a composite

Thank you for your attention! ^__^

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