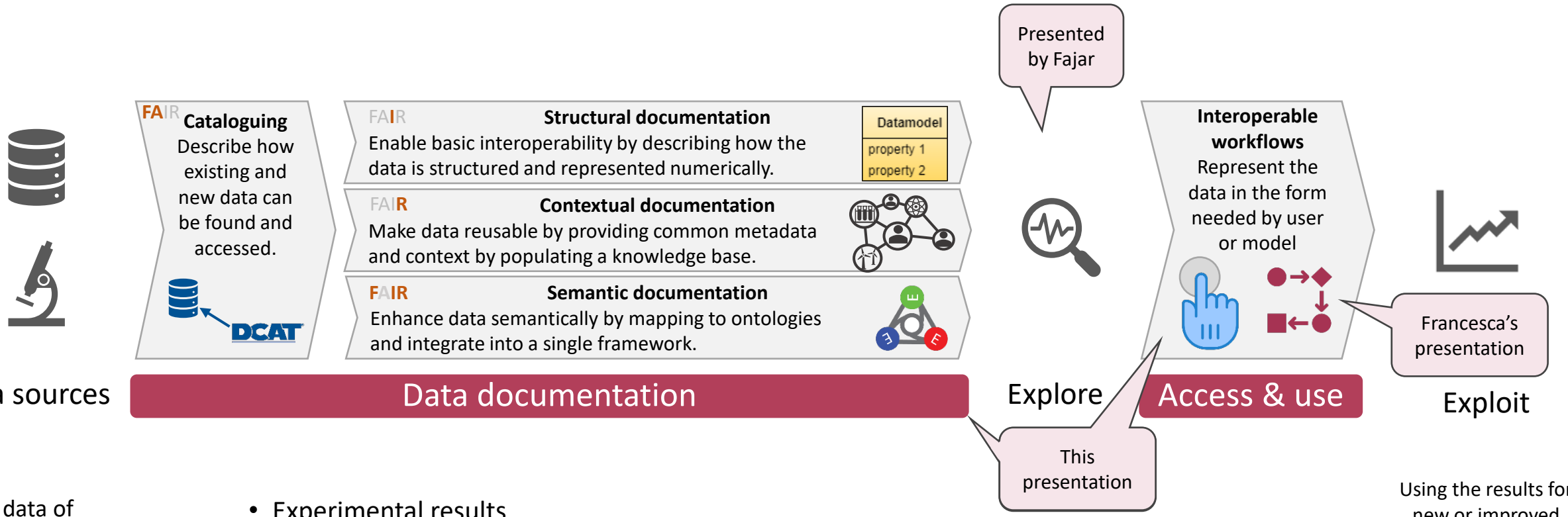
OntoTrans

Connecting data consumers to data sources

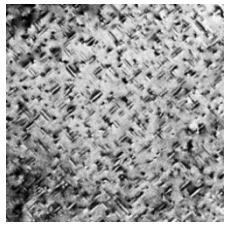
Demonstration OTEAPI and OTELib pipelines

Jesper Friis, SINTEF

How to represent a user case in practice?



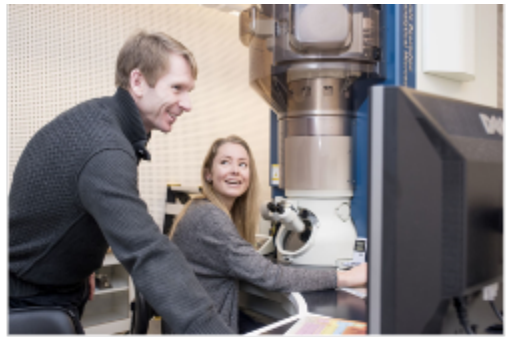
TEM data of aluminium

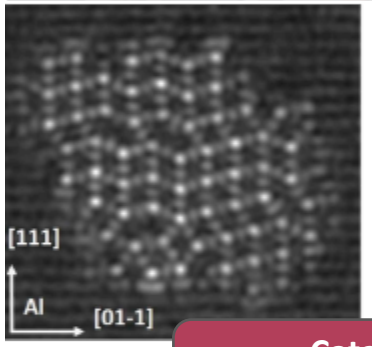


- Experimental results
- Available models and their input and output
- Alloy composition
- Thermo-mechanical processing of the alloy
- Microstructure (at different processing steps)
- Material properties

Using the results for new or improved products

Data documentation with OTEAPI





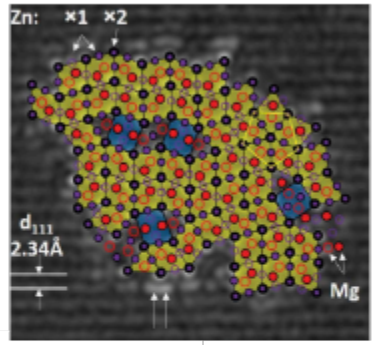
Structural documentation
Datamodel

Input datamodel
property 1
property 2

Semantic documentation
Ontological mappings

Contextual documentation
Knowledge base

Wanted datamodel
property 1
property 2



Access → **Parse** → **Map** → **Map** → **Generate**

Partial pipeline documenting the data source

Partial pipeline documenting how you want the data

g(f(x))
Mapping function

Cataloguing
dcat:downloadUrl, …

Data Access
Accessing information from data-services or by downloading files

Syntactic Analysis
Read or parse specific formats into an instance of the input data model.

Semantic Mapping
Associate data or properties in data models with ontological concepts.

Syntactic representation
Serialise instance of wanted data model to a specific format.

Demo

GitHub: https://github.com/SINTEF/dlite/tree/tem_example/examples/TEM_data

dlite / examples / TEM_data / ↑ Top

README.md

TEM data Example

This example demonstrates how a user can access and combine different datasets and represent them in the form he or she needs.

The basic idea is to semantically document both the datasets and the different ways one want to represent them as.

The example uses DLite as the underlying interoperability framework, but also introduces a set of other tools for working with semantically documented data.

Content

Demo

Implemented in [demo.py]

Accessing the knowledge base (created and populate a local one in this case)

```
ts = Triplestore(backend="rdflib")
populate_triplestore(ts, indir / "resources.yaml")
```

Defining some handy namespace prefixes

```
OTEIO = Namespace("http://emmo.info/oteio#")
PM = ts.bind("pm", "https://www.ntnu.edu/physmet/data#")
```

Used technologies

OTEAPI – data documentation
https://github.com/EMMC-ASBL/oteapi-core/

TRIPPER

Tripper – Common interface to triplestores
https://github.com/EMMC-ASBL/tripper/

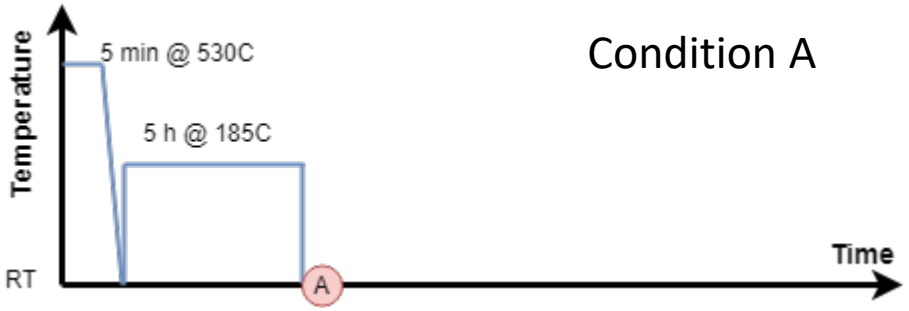
DLite

DLite – interoperability framework
https://github.com/SINTEF/dlite/

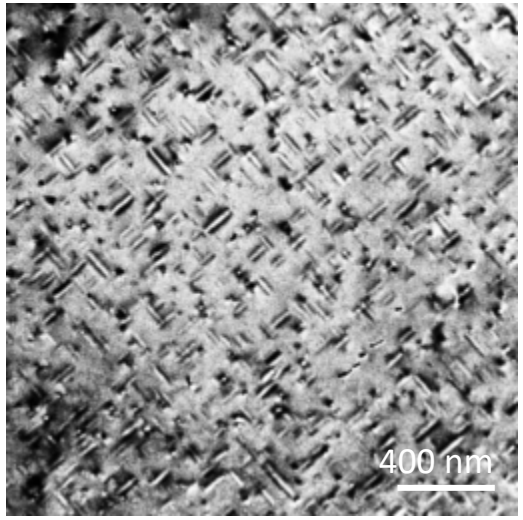
The user case

Alloy	Al	Mg	Si	Mn	Fe
Al-Mg-Si	97.88	0.8	0.8	0.5	0.02

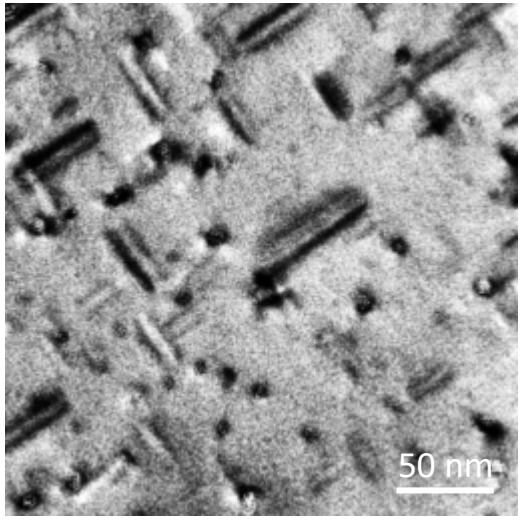
Aim: understand how the alloy composition and heat treatment influences the microstructure and alloy properties. Use this understanding to develop new alloys with improved or tailored properties.



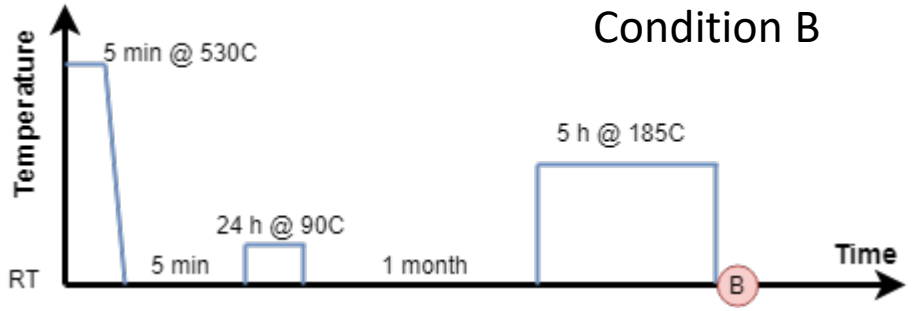
Condition A



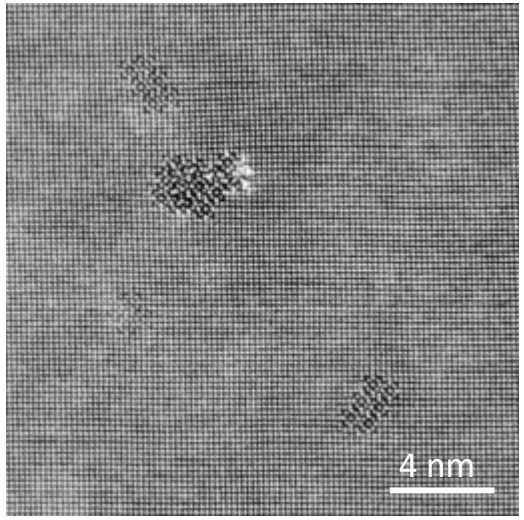
pm:TEM_BF_100-at-m5-and-2_001



pm:TEM_040



Condition B



pm:TEM_6c8cm_008

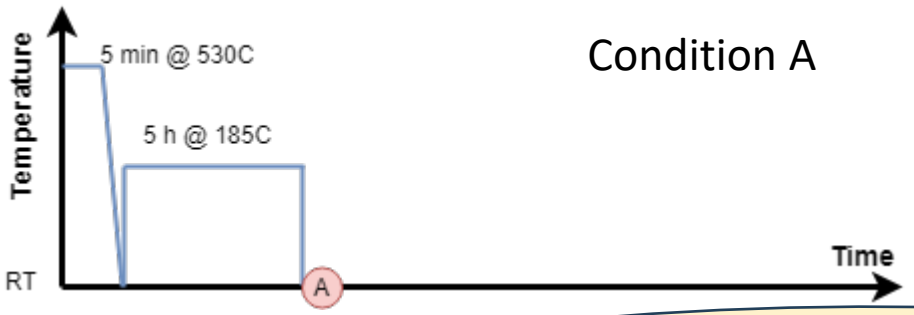
How do I document my data?

Demo steps

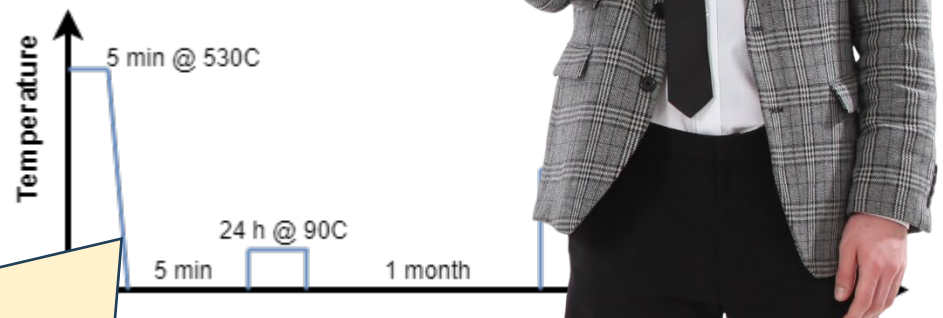
Alloy	Al	Mg	Si	Fe
Al-Mg-Si	97.88	0.8		

How do I see what datasets are available?

Aim: understand how the a… …s the microstructur… …s.
Use this understanding to develop new alloys with improved or tailored properties.

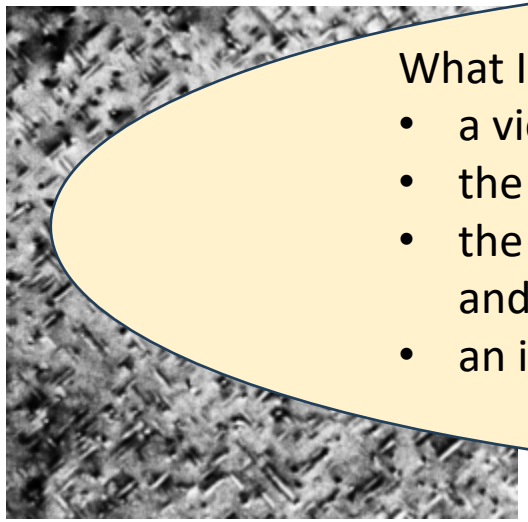


Condition A

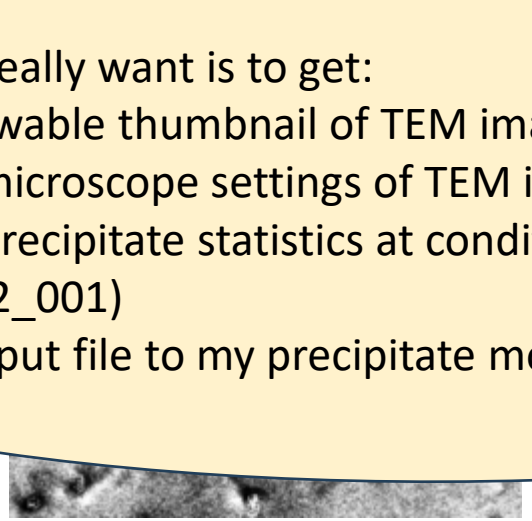


What I really want is to get:

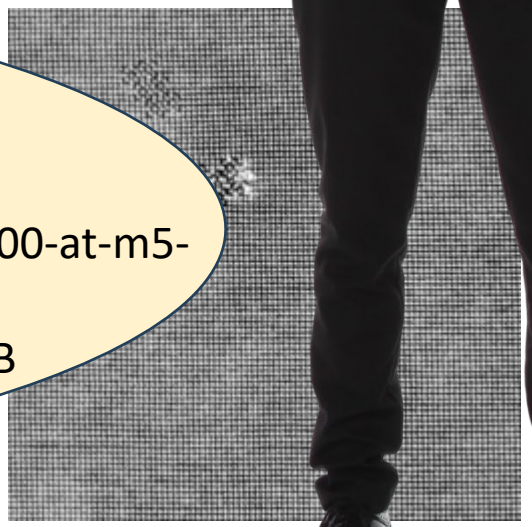
- a viewable thumbnail of TEM image pm:TEM_6c8cm_008
- the microscope settings of TEM image pm:TEM_040
- the precipitate statistics at condition A (based on pm:TEM_BF_100-at-m5-and-2_001)
- an input file to my precipitate model for my alloy and condition B



pm:TEM_BF_100-at-m5-and-2_001



pm:TEM_040



pm:TEM_6c8cm_008

Document a TEM image

YAML representation

IRI of individual standing for the image

Where the data can be obtained

DLite storage plugin to use for parsing

Local label referring to this data instance

Prefix for TEMImage datamodel

Mapping properties in the datamodel to ontological concepts

```yaml
https://www.ntnu.edu/physmet/data#TEM_6c8cm_008:
  dataresource:
    downloadUrl: https://folk.ntnu.no/friisj/temdata/6c8cm_008.dm3
    mediaType: application/vnd.dlite-parse
    configuration:
      driver: dm3
      label: temimage

  mapping:
    mappingType: mappings
    prefixes:
      temimage: http://onto-ns.com/meta/0.1/TEMImage#
      map: http://emmo.info/domain-mappings#
      emmo: http://emmo.info/emmo#
      oteio: http://emmo.info/oteio#
    triples:
      - [temimage:filename,   map:mapsTo,  oteio.FileName]
      - [temimage:data,       map:mapsTo,  emmo.Array]
      - [temimage:pixelUnit,  map:mapsTo,  emmo.Unit]
      - [temimage:pixelSize,  map:mapsTo,  emmo.Length]
      - [temimage:metadata,   map:mapsTo,  oteio.Dictionary]
```

TEMImage datamodel

```json
"uri": "http://onto-ns.com/meta/0.1/TEMImage",
"description": "Metadata for a TEM image.",
"dimensions": {
  "ndim": "Number of dimensions. This is always 4.",
  "zSize2": "Number of pixels along the 4th dimension for a 4D file (e.g. SPED).",
  "zSize": "Number of pixels along the 3rd dimension for a 3D file.",
  "ySize": "Number of pixels along the 2nd dimension (y-axis).",
  "xSize": "Number of pixels along the 1st dimension (x-axis)."
},
"properties": {
  "filename": {
    "type": "string",
    "description": "File name."
  },
  "data": {
    "type": "float64",
    "shape": ["zSize2", "zSize", "ySize", "xSize"],
    "description": "Image data. Note that the indexing starts with the 4th dimension. For
  },
  "pixelUnit": {
    "type": "string",
    "shape": ["ndim"],
    "description": "The unit name as a string for each dimension of each dataset."
  },
  "pixelSize": {
    "type": "float64",
    "shape": ["ndim"],
    "description": "The real size of the pixel. Real and reciprical space are supported."
  },
  "metadata": {
    "type": "string",
    "description": "A json dump of relevant metadata tags."
  }
}
}
```

Explore

List all documented data sources and sinks

```
# List all data sources and sinks
>>> print("Data sources:")
>>> for source in ts.subjects(RDF.type, OTEIO.DataSource):
...     print("  -", source)
>>> print()
>>> print("Data sinks:")
>>> for sink in ts.subjects(RDF.type, OTEIO.DataSink):
...     print("  -", sink)
Data sources:
  - https://www.ntnu.edu/physmet/data#TEM_BF_100-at-m5-and-2_001
  - https://www.ntnu.edu/physmet/data#TEM_040
  - https://www.ntnu.edu/physmet/data#TEM_6c8cm_008
  - https://www.ntnu.edu/physmet/data#alloy_composition

Data sinks:
  - https://www.ntnu.edu/physmet/data#thumbnail_image
  - https://www.ntnu.edu/physmet/data#microscope_settings
  - https://www.ntnu.edu/physmet/data#image_analyser
  - https://www.ntnu.edu/physmet/data#precipitate_statistics
  - https://www.ntnu.edu/physmet/data#precipitation_model_input
```

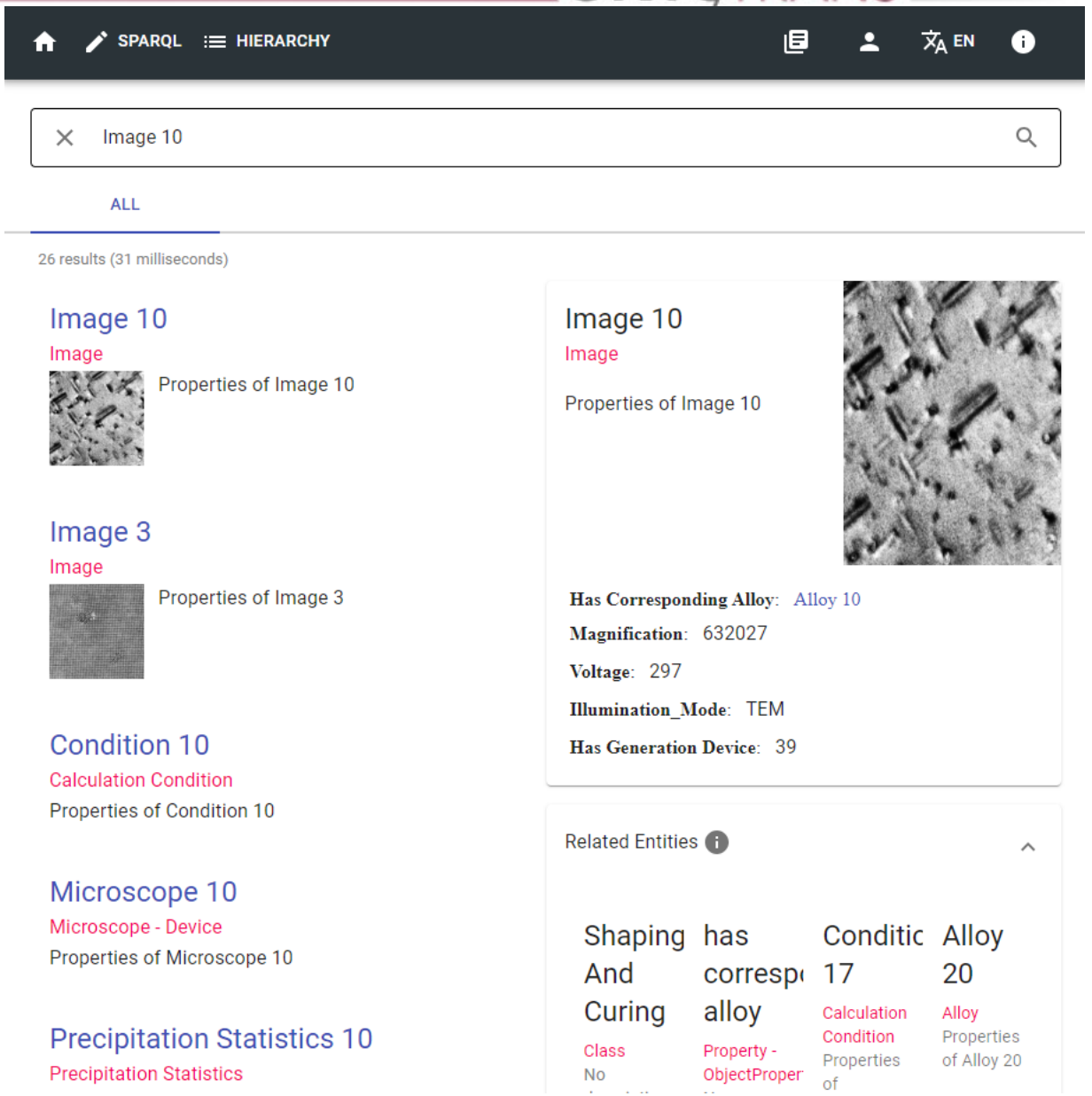
Explore

Note:
The ESS provides a wonderful web gui for data exploration

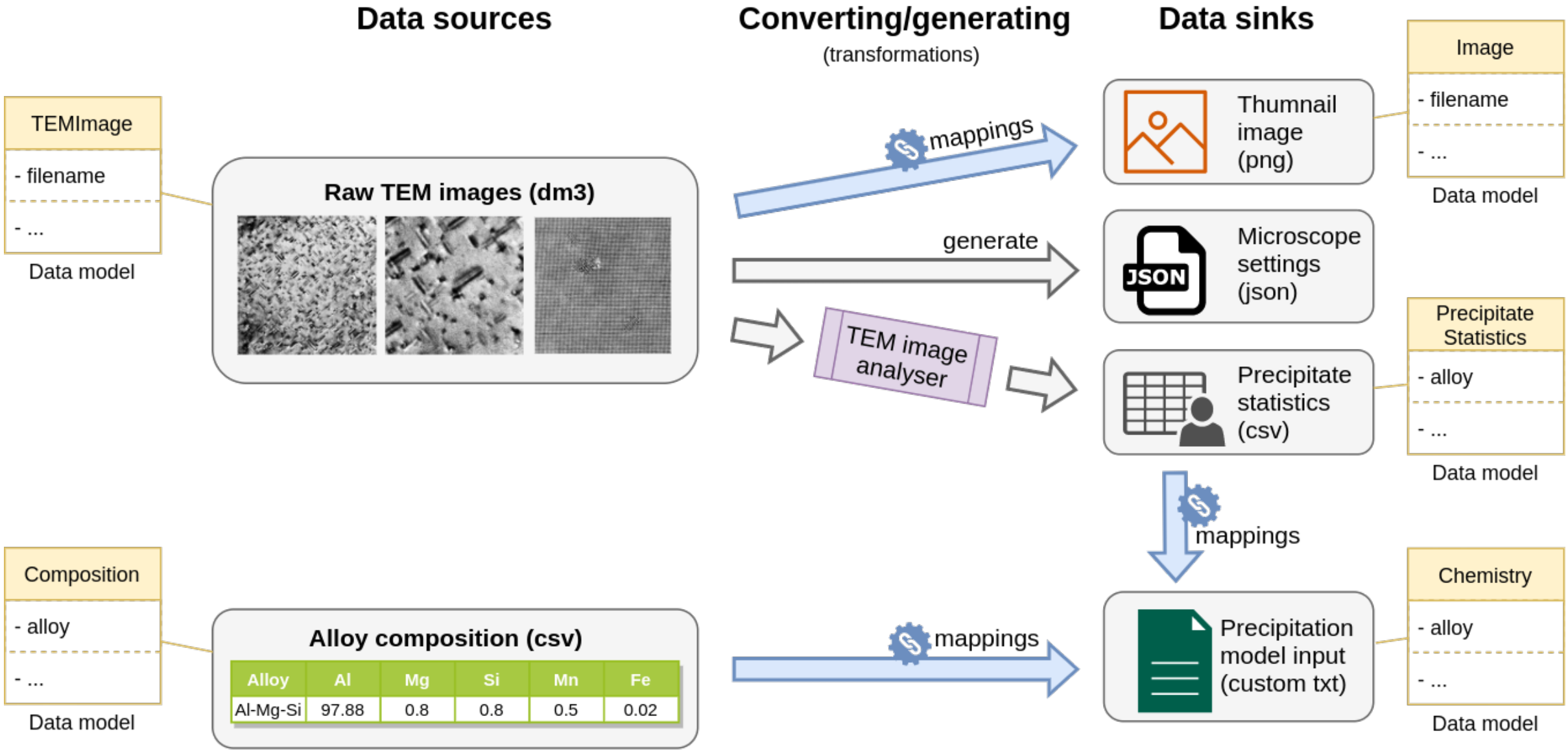
List all documented data sources and sinks

```
# List all data sources and sinks
>>> print("Data sources:")
>>> for source in ts.subjects(RDF.type, OTEIO.DataSource):
...     print("  -", source)
>>> print()
>>> print("Data sinks:")
>>> for sink in ts.subjects(RDF.type, OTEIO.DataSink):
...     print("  -", sink)
Data sources:
  - https://www.ntnu.edu/physmet/data#TEM_BF_100-at-m5-and-2_001
  - https://www.ntnu.edu/physmet/data#TEM_040
  - https://www.ntnu.edu/physmet/data#TEM_6c8cm_008
  - https://www.ntnu.edu/physmet/data#alloy_composition

Data sinks:
  - https://www.ntnu.edu/physmet/data#thumbnail_image
  - https://www.ntnu.edu/physmet/data#microscope_settings
  - https://www.ntnu.edu/physmet/data#image_analyser
  - https://www.ntnu.edu/physmet/data#precipitate_statistics
  - https://www.ntnu.edu/physmet/data#precipitation_model_input
```



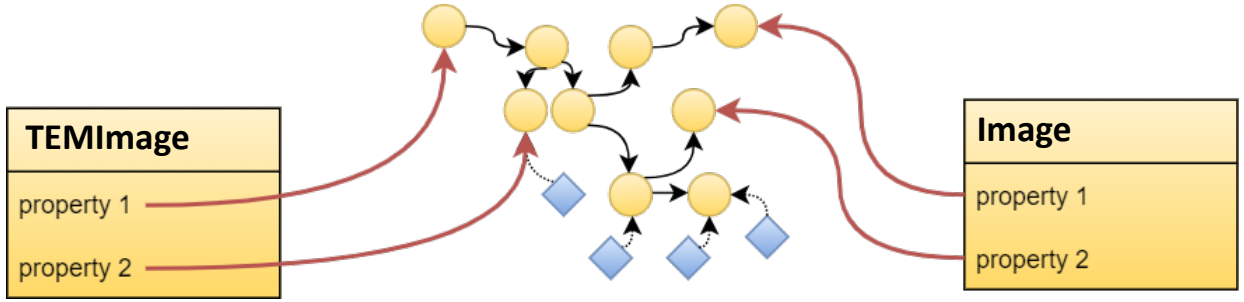
Accessing datasets



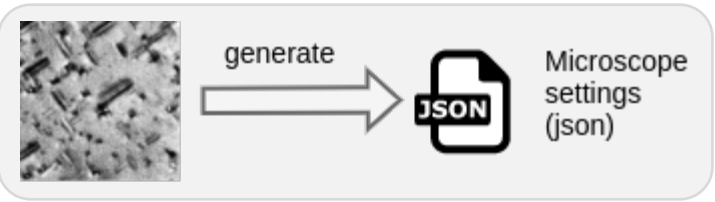
Accessing datasets
thumbnail image

Get a thumbnail of TEM image pm:TEM_6c8cm_008

```
get_data(ts, steps=(PM.TEM_6c8cm_008, PM.thumbnail_image))
```

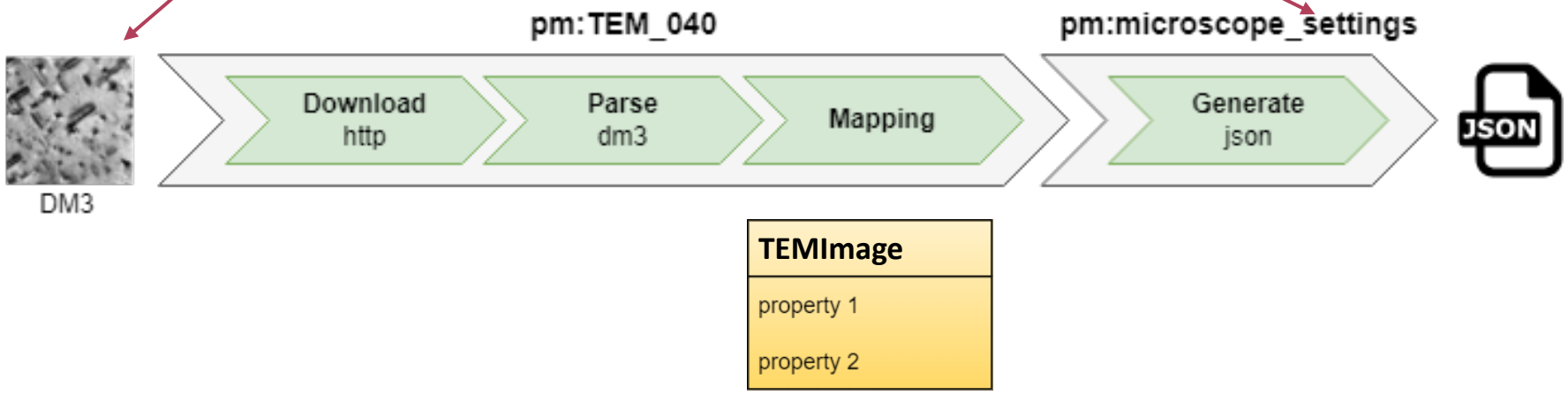


Accessing datasets
microscope settings

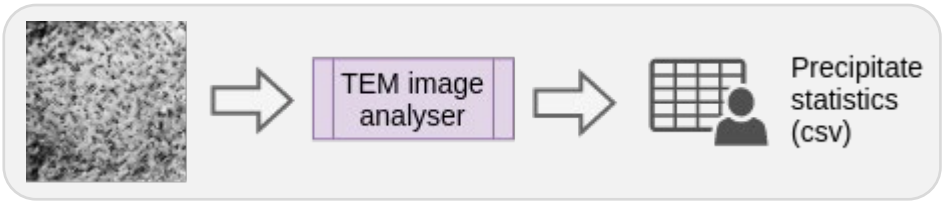


Get microscope setting for TEM image pm:TEM_6c8cm_008

```
get_data(ts, steps=(PM.TEM_040, PM.microscope_settings))
```

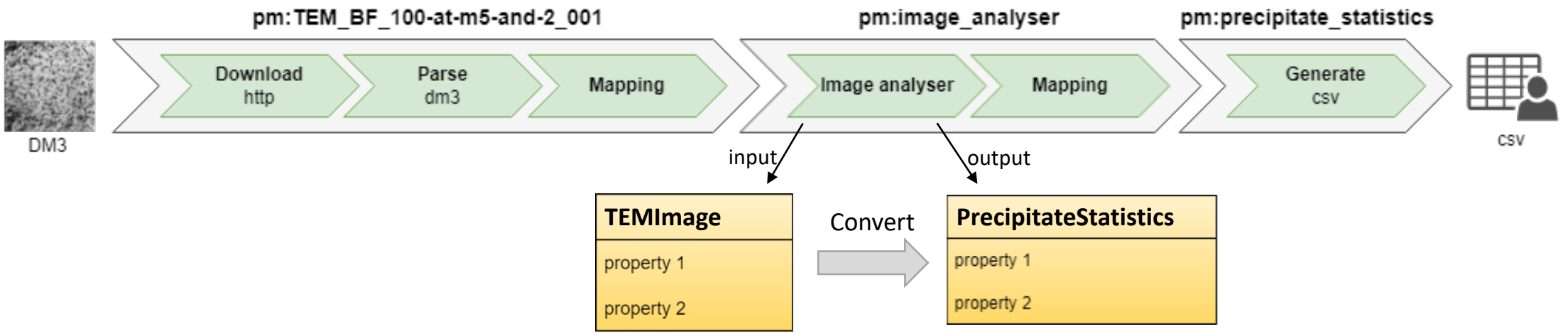


Accessing datasets
precipitate statistics

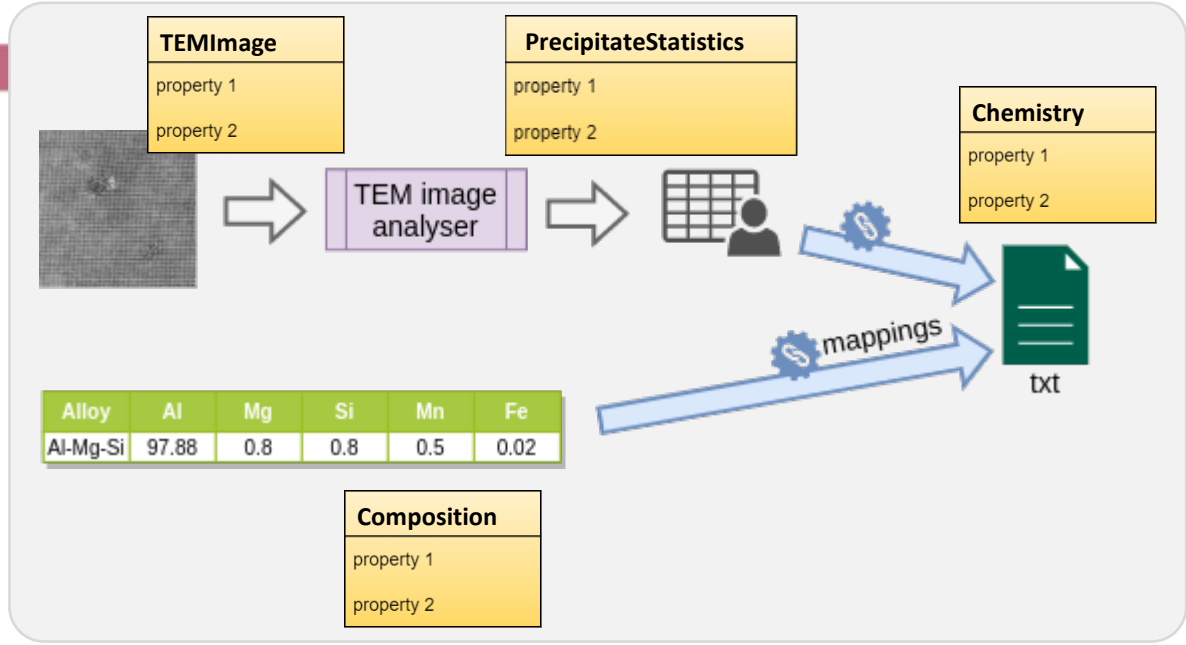


Get precipitate statistics for TEM image pm:TEM_6c8cm_008

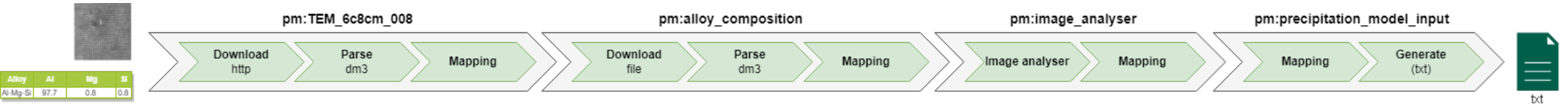
```
get_data(ts, steps=(
    PM["TEM_BF_100-at-m5-and-2_001"],  # TEM image to analyse
    PM.image_analyser,                 # Postprocess with the image analyser
    PM.precipitate_statistics,         # How to present the result
))
```



Accessing datasets
precipitation model input file

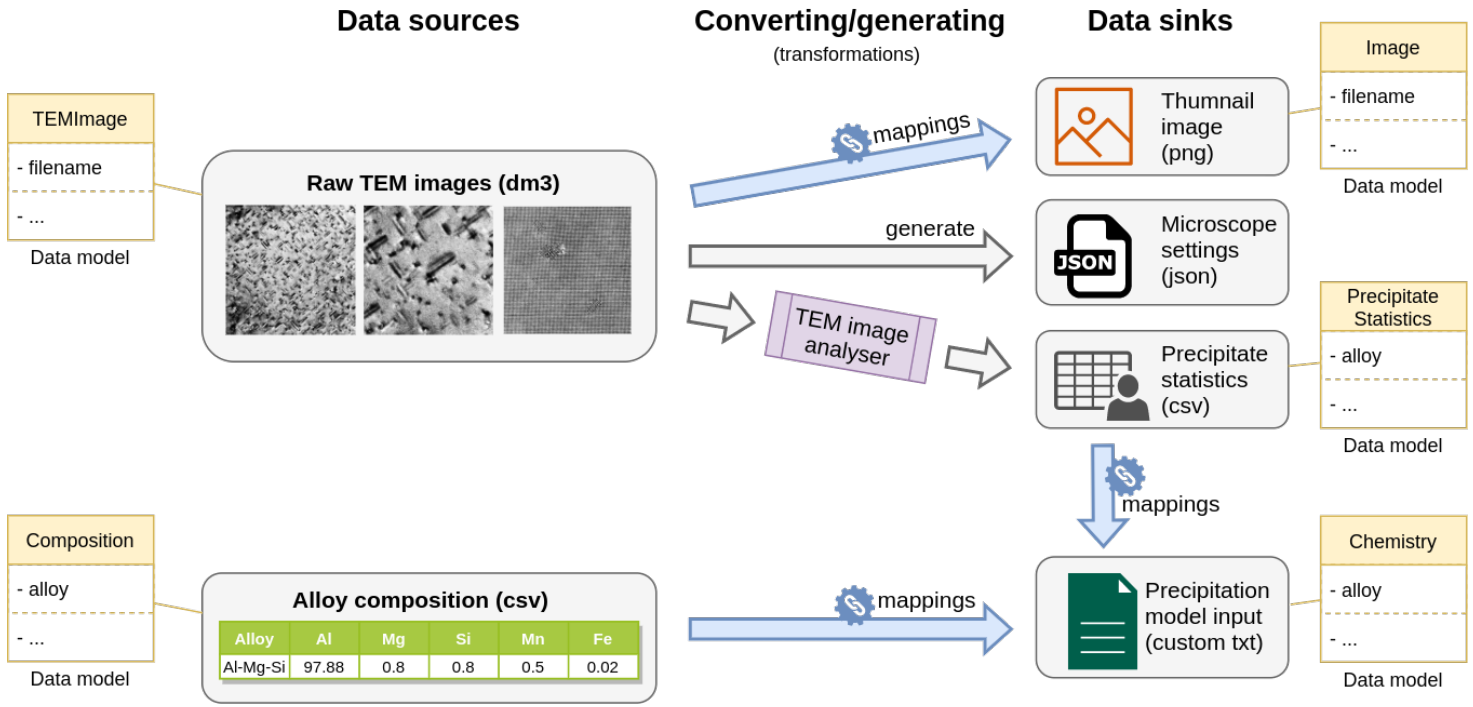



```
# Get input to presipitation model based on the combination of the
# alloy composition and precipitation statistics obtained from
# postprocessing TEM image pm:TEM_6c8cm_008.
get_data(ts, steps=(
    PM.alloy_composition,          # Data source 1: alloy composition
    PM.TEM_6c8cm_008,              # Data source 2: TEM image
    PM.image_analyser,             # Postprocess with the image analyser
    PM.precipitation_model_input,  # How to present the result
))
```



Summary

- We have a simple Python interface for connecting data consumers to data sources
- Demonstrated on a user case on aluminium

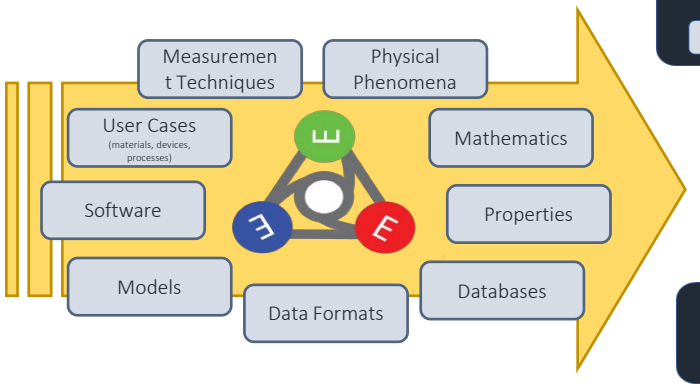


Summary

- We have a simple Python interface for connecting data consumers to data sources
- Demonstrated on a user case on aluminium

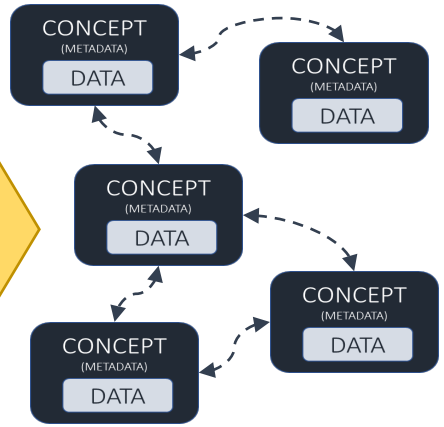
USER CASE

From real world entities at different scales…



ONTOLOGY

…through a formal knowledge-based representational system…



KNOWLEDGE

…to a digital representation and knowledge management.

Thank you!

The OntoTrans project has received funding from the European Union‘s Horizon 2020 research and innovation programme under Grant Agreement No 862136.