INTERFACING ROXIE

MATTHIAS BONORA

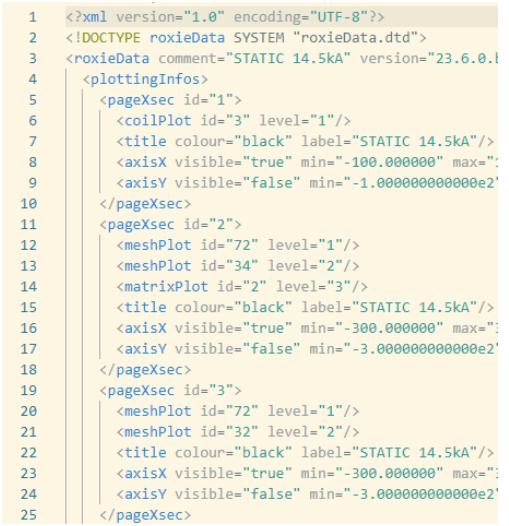
MT-28
International Conference
on Magnet Technology
Aix-en-Provence, France, 10-15 September 2023

Overview

- Interfaces from ROXIE
  - xml, hmo, geometry
  - Interfacing during execution
- Scripting ROXIE
  - Python API, interfacing API
- Working with Systems and Models (MBSE)
  - pyMBSE, MMBSE, developments

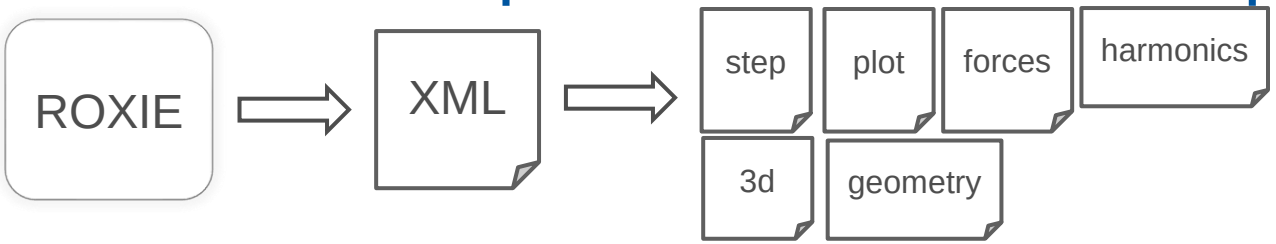
Interfaces from ROXIE

- External mesh input (.hmo)
- Simplify ROXIE output structure
- ~~.output, .post, .force2D, .map2d, .wrl~~
- -> 2 main outputs
  - .pdf for Report
  - .xml for structured output
  - Contains plots, graphs, geometry, dv, objectives,…

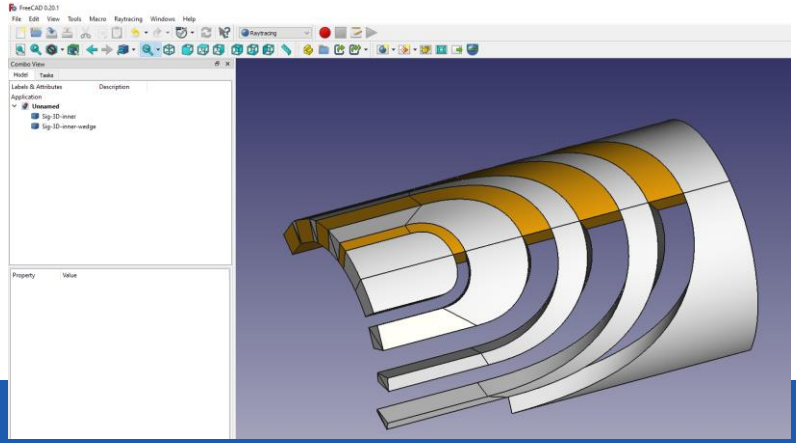


Interfaces from ROXIE

- Extension to specific format via scripts

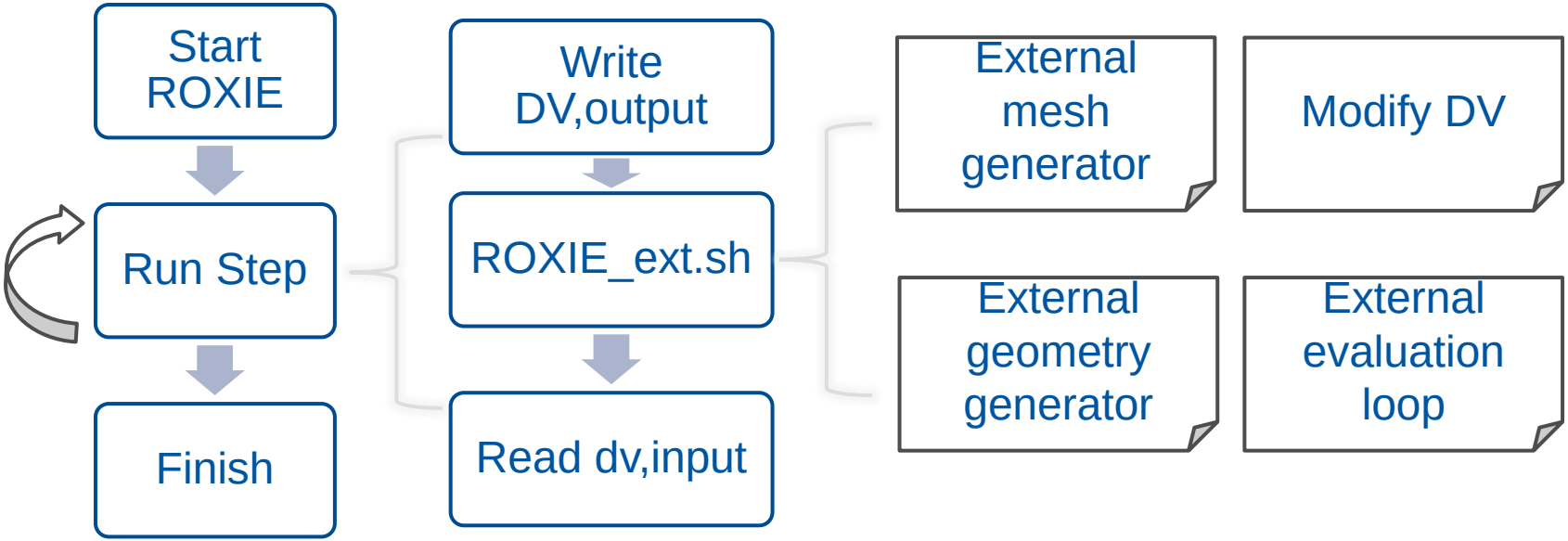


- Standards for output format



Interfaces from ROXIE

- Interfacing during execution (WIP)



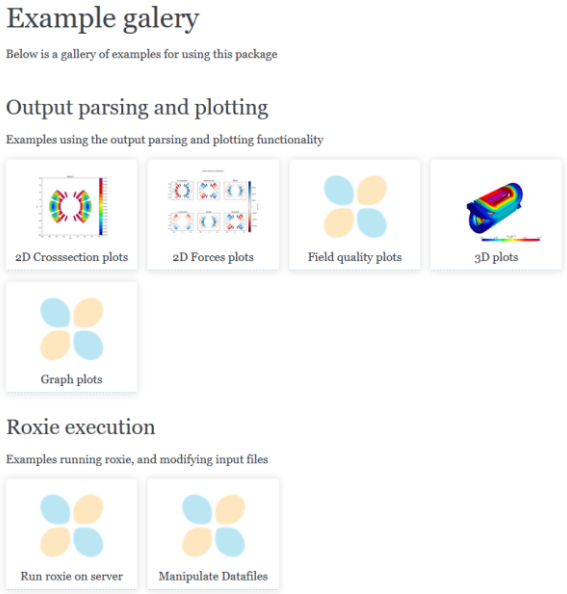
Scripting ROXIE

- Extend ROXIE functionality in Python
- ROXIE-api: interact with ROXIE (execution, xml outputs, plots, input file generation)

- ROXIE-interfaces: generate & transform files to use in ROXIE (.hmo, meshes, geometry)

Scripting ROXIE – ROXIE-api

- Documentation
- Handle .data files
  - Load existing models
  - Transform tables, set flags
  - Generate new data files
- Handle xml output files
  - Plot
  - Extract data
- Execute ROXIE



Sripting ROXIE – input files

- Load existing files
- Operate on tables / flags
- Copy between Files
- Save as new File

Setting up

Load Datafile into RoxieInputBuilder object

```
import pathlib
import tempfile
from roxieapi.input.builder import RoxieInputBuilder

folder = pathlib.Path("../input_files").absolute()
datafile = "dipole_2d.data"

rib = RoxieInputBuilder.from_datafile(folder / datafile)
```

Change flags and metadata

```
rib.comment = "Python modified datafile"
rib.flags["LDEBUG"] = True
rib.flags["LFORCE2D"] = True

rib.flags
```

Check defined Blocks

```
rib.block
```

Update alpha angle of block no 4

```
# Dataframe index 3 corresponds to Block no 4
rib.block.loc[3, "alpha"] = 23.2

rib.block
```

Generate new datafile

```
tmp_dir = pathlib.Path(tempfile.gettempdir())
tmp_file = tmp_dir / "temp.data"
rib.build(str(tmp_file))

with open(tmp_file, "r") as f:
    print(f.read())
```

Scripting ROXIE - execution

- Simple Upload / execution
- Return of output log, status
- Download of results
- Run locally (terminal)
- OR remotely via REST
  - Server, Docker

Executing roxie

Set up paths, datafile and execute Roxie as a process

```
import pathlib
from roxieapi.tool_adapter.RoxieToolAdapter import TerminalRoxieToolAdapter

folder = pathlib.Path("../input_files").absolute()
datafile = "dipole_2d.data"

runner = TerminalRoxieToolAdapter(input_file=datafile, input_folder_rel_dir=folder)

result = runner.run()

print(f"Run executed with return code {result}")
```

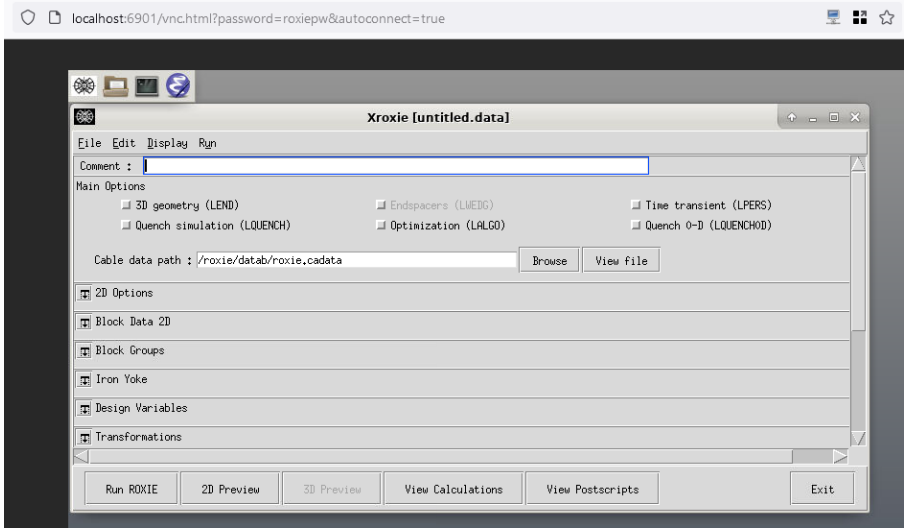
Check output and errors

```
for line in runner.output_lines:
    print(line)
```

```
17  # Executing roxie
18  # ----------
19  # Set up paths, datafile and execute Roxie as a process
20  #
21  import pathlib
22  from roxieapi.tool_adapter.RoxieToolAdapter import RestRoxieToolAdapter
23
24  folder = pathlib.Path("../input_files").absolute()
25  datafile = "dipole_2d.data"
26
27  runner = RestRoxieToolAdapter(folder, datafile, "my-model", "localhost", 8080)
28
29  result = runner.run()
30
31  print(f"Run executed with return code {result}")
32
    Run Cell | Run Above | Debug Cell
33  # %%
34  # Check output and errors
35  # -------------
36
37  for line in runner.output_lines:
38      print(line)
39
```

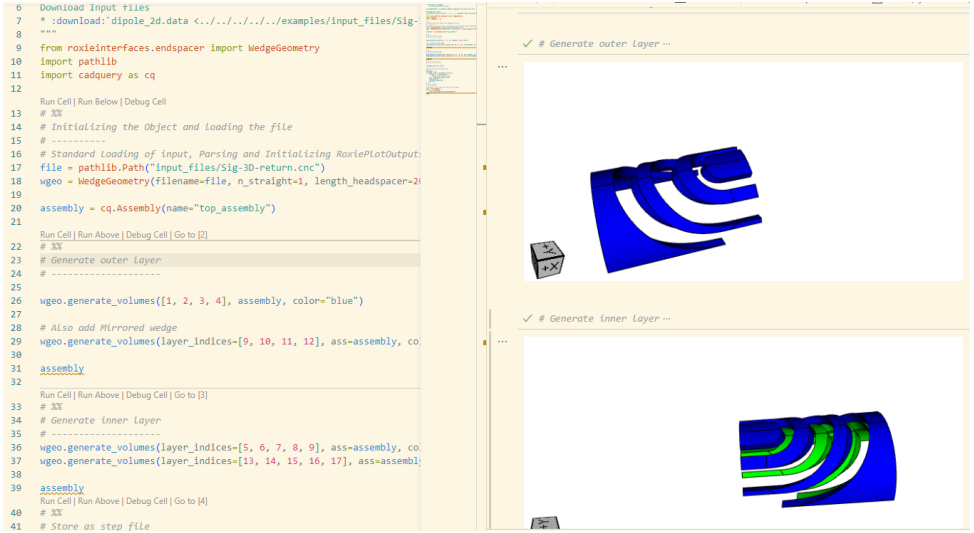
ROXIE in Docker

- ROXIE Docker container (see doc)
  - Without Linux server
  - from any system*
  - VNC Server/client
  - REST API for python



ROXIE Interfaces package

- Documentation
- Collection of scripts for translating IO files
  - ROXIE to STEP (wedges, coils)
  - Hypermesh hmascii to hmo
  - Gmsh to hmo (WIP)

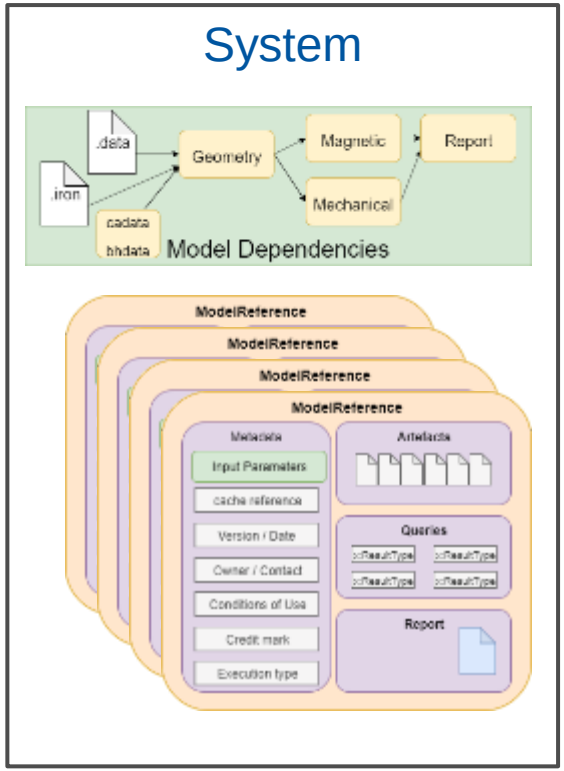


Linking everything together

pyMBSE, MMBSE, Magnet Exchange

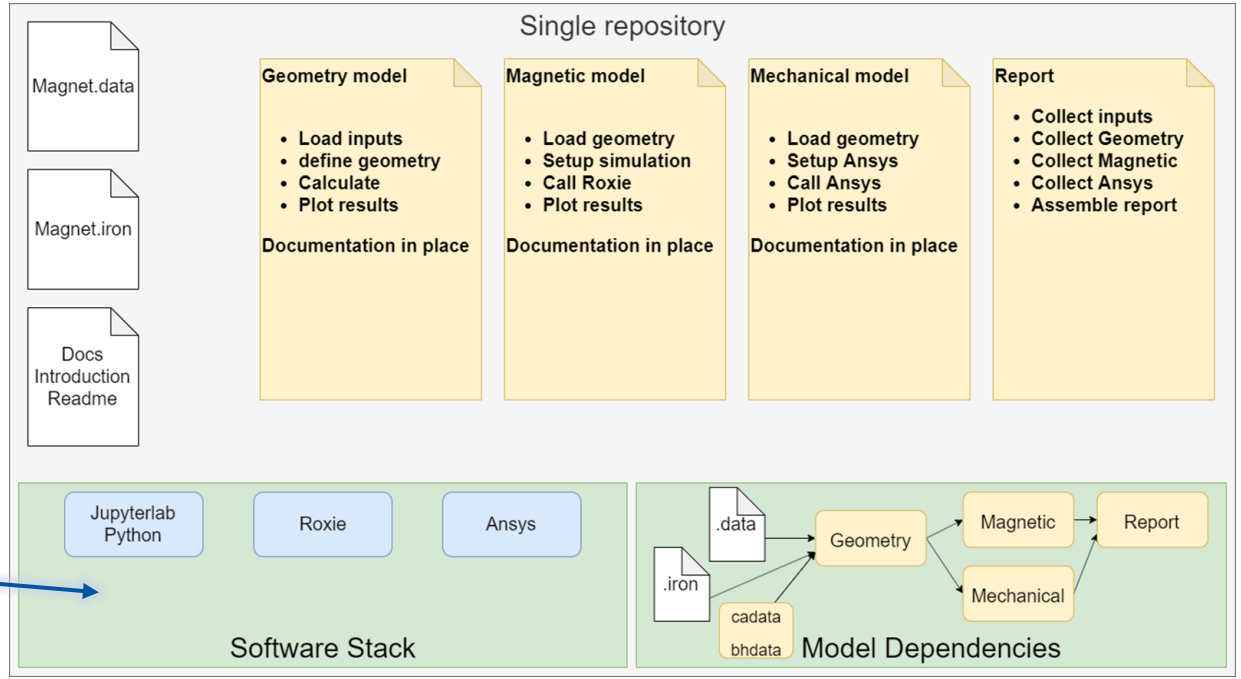
Linking everything together - pyMBSE

- So far: Small modules, packages and projects on top of ROXIE
  - R&D of features and extensions
  - Missing Concept of Integration
- → pyMBSE
  - Started in PSI for the CHART/MagNum project
  - Introduce the concept of models, interfaces and model based system engineering to magnet design
  - Define Magnet Models, dependencies, inputs, outputs, Dependency graph
  - Link with Measurements
  - **Integrated Environment**
    - Software
    - Tools
    - Packages

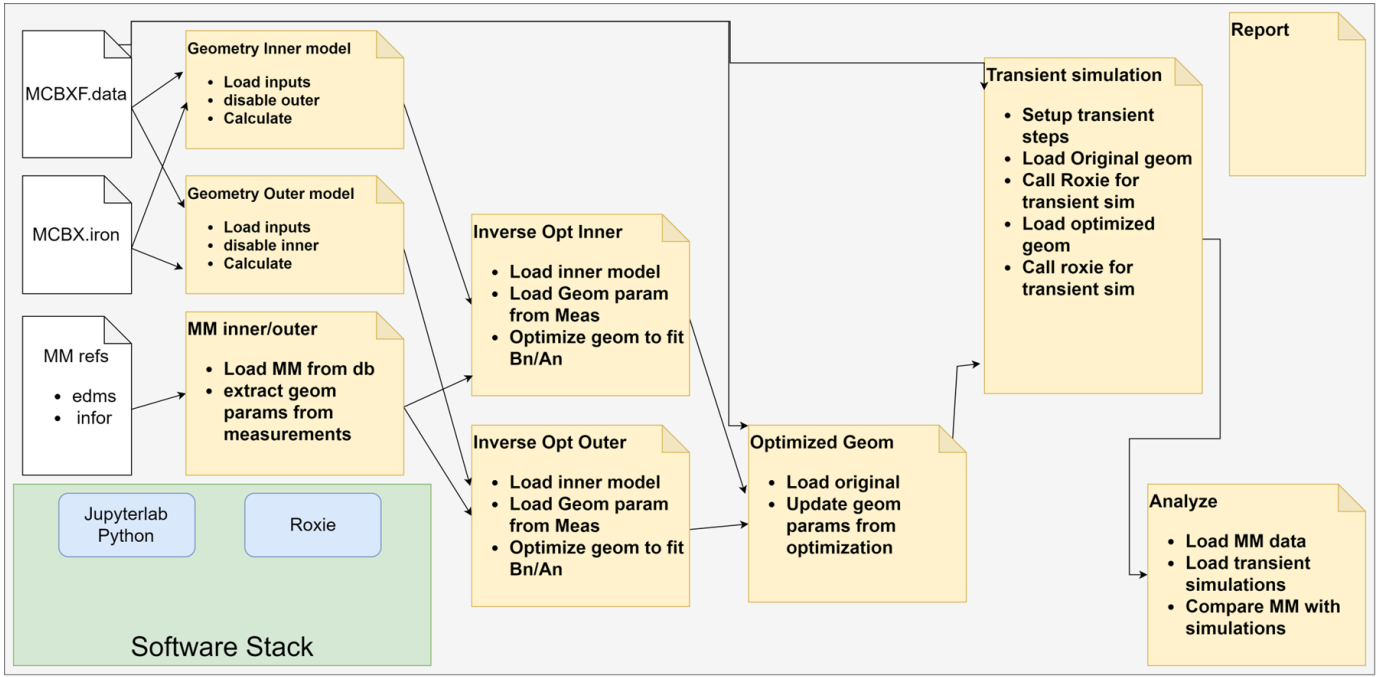


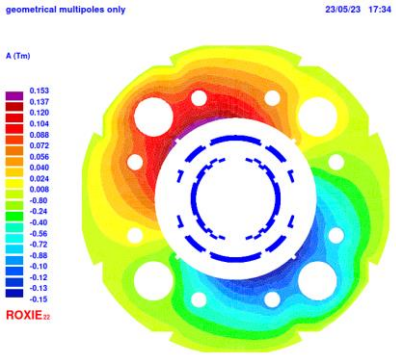
Linking everything together - pyMBSE

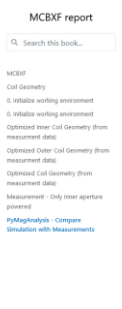
Docker containers

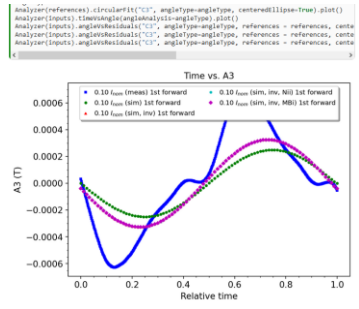


Linking everything together – pyMBSE Example: MCBXF





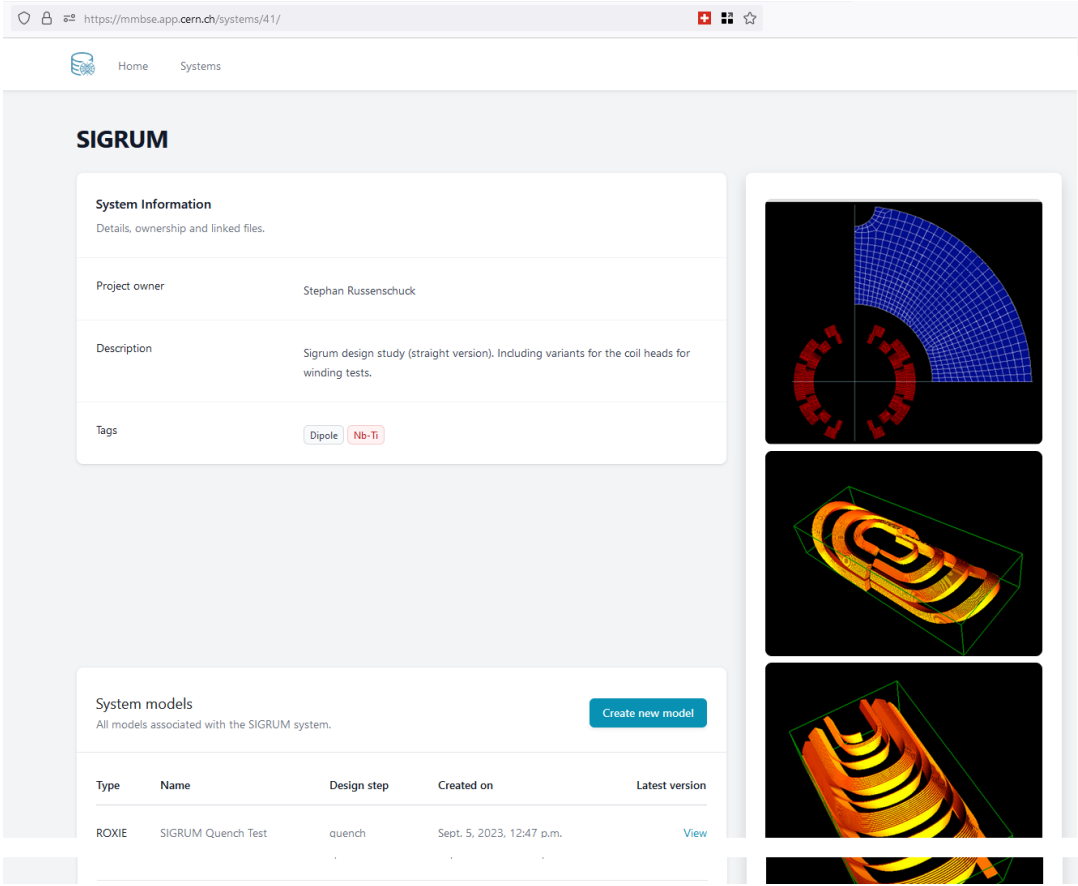




MMBSE

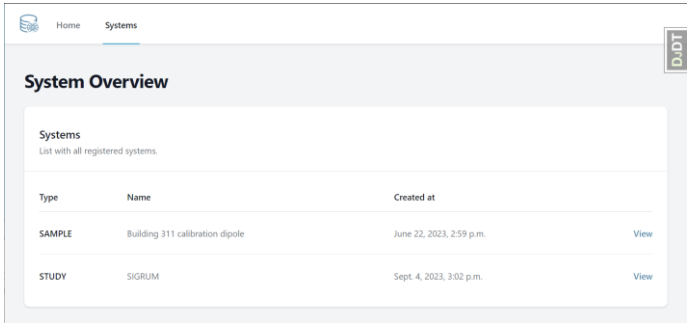
- Database with Magnets and systems
- Different Models
- REST API
  - Scriptable

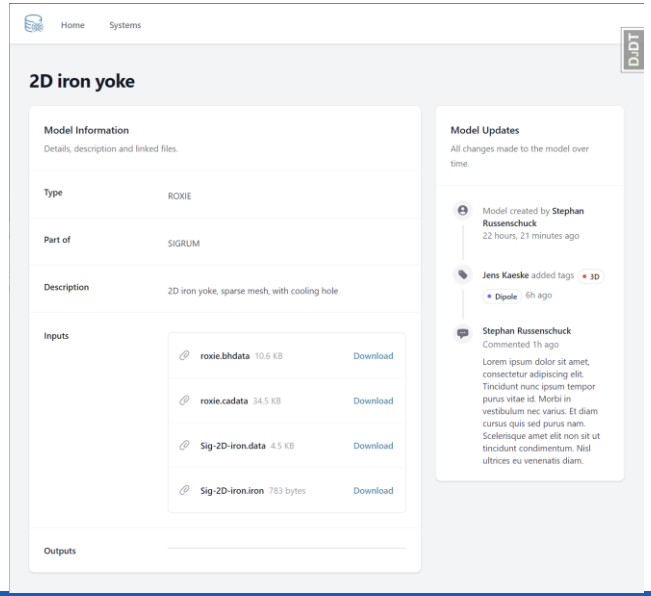
- Work in Progress
  - Connect to ROXIE GUI
  - Connect with pyMBSE



MMBSE

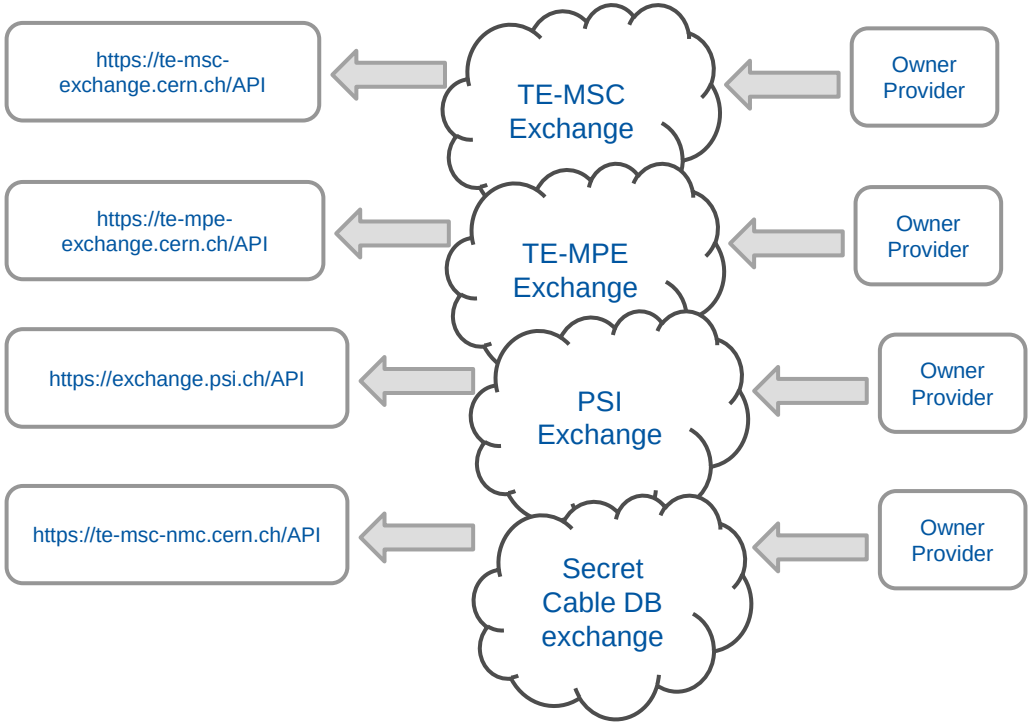
- Store Systems
- Store Inputs
  - Files
  - Dependencies to other Models
  - Parameters
- Store Outputs of execution
  - Version
  - Report
  - Result files
- Access via API (Python)
  - Connect to other System



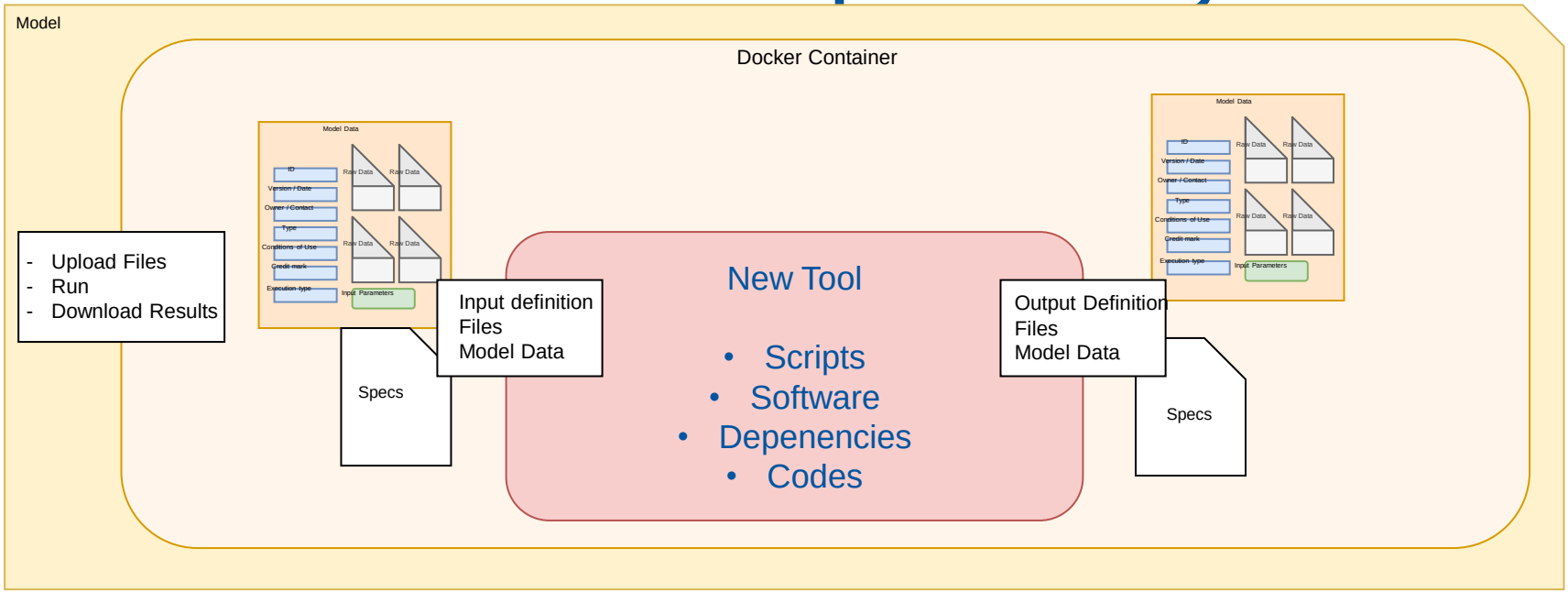


MMBSE - Boundary crossing

- Separate by responsibility,
- and End points
- Host seperately
  - Access restrictions
  - Data restrictions
  - Computing Resources



MMBSE - Developer Story



Summary

- New Options to extend ROXIE
- Methodology to combine design models
- Environment for Sustainable Magnet development