Metadata Crawling for (HPMC) Simulations

Archetype DORIS: High Performance Measurement & Computing (HPMC)

Technical University of Munich (TUM)
Giuseppe **Chiapparino** | giuseppe.chiapparino@tum.de
Benjamin **Farnbacher** | benjamin.farnbacher@tum.de
Chair of Aerodynamics and Fluid Mechanics

High Performance Computing Center Stuttgart (HLRS)

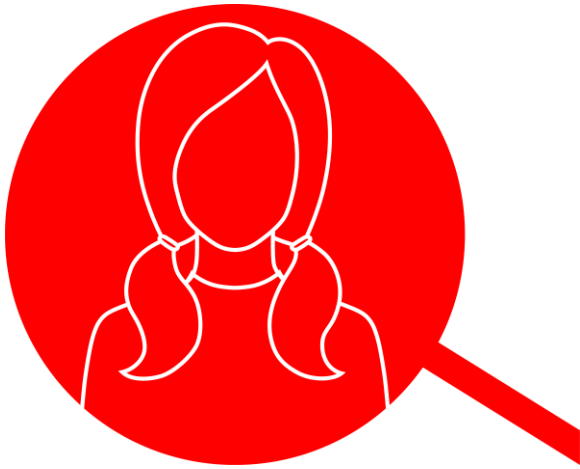
Leibniz Supercomputing Centre (LRZ)

RWTH Aachen University (AIA)

Task Area DORIS: HPMC

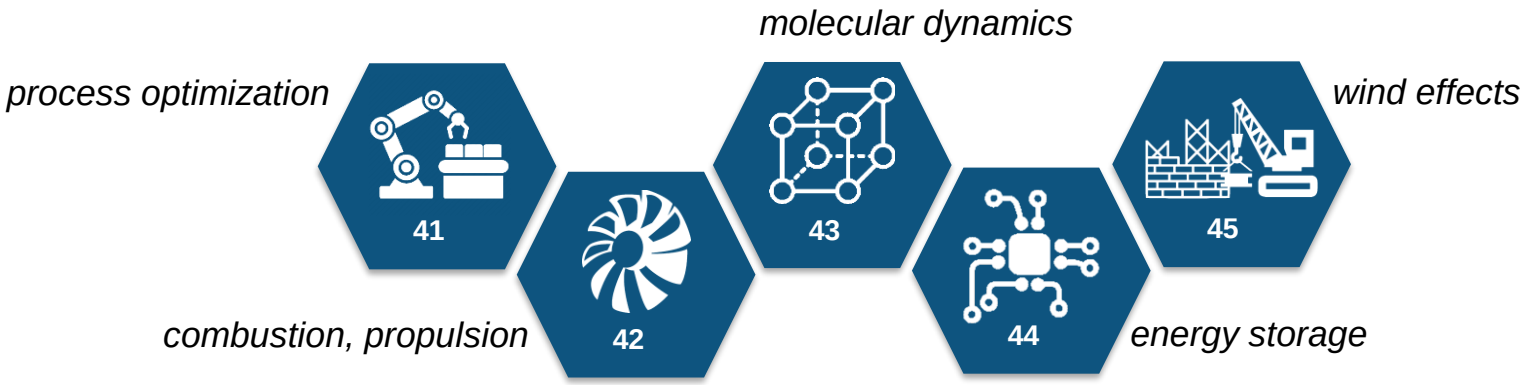
*… I'm an engineer conducting and post-processing high-resolution and **high-performance measurements and computation** (simulation) **with very large data** on HPC systems.*

The data sets I work with are extremely large and as such are largely immobile. This mandates tailored, hand-made software."



My needs are

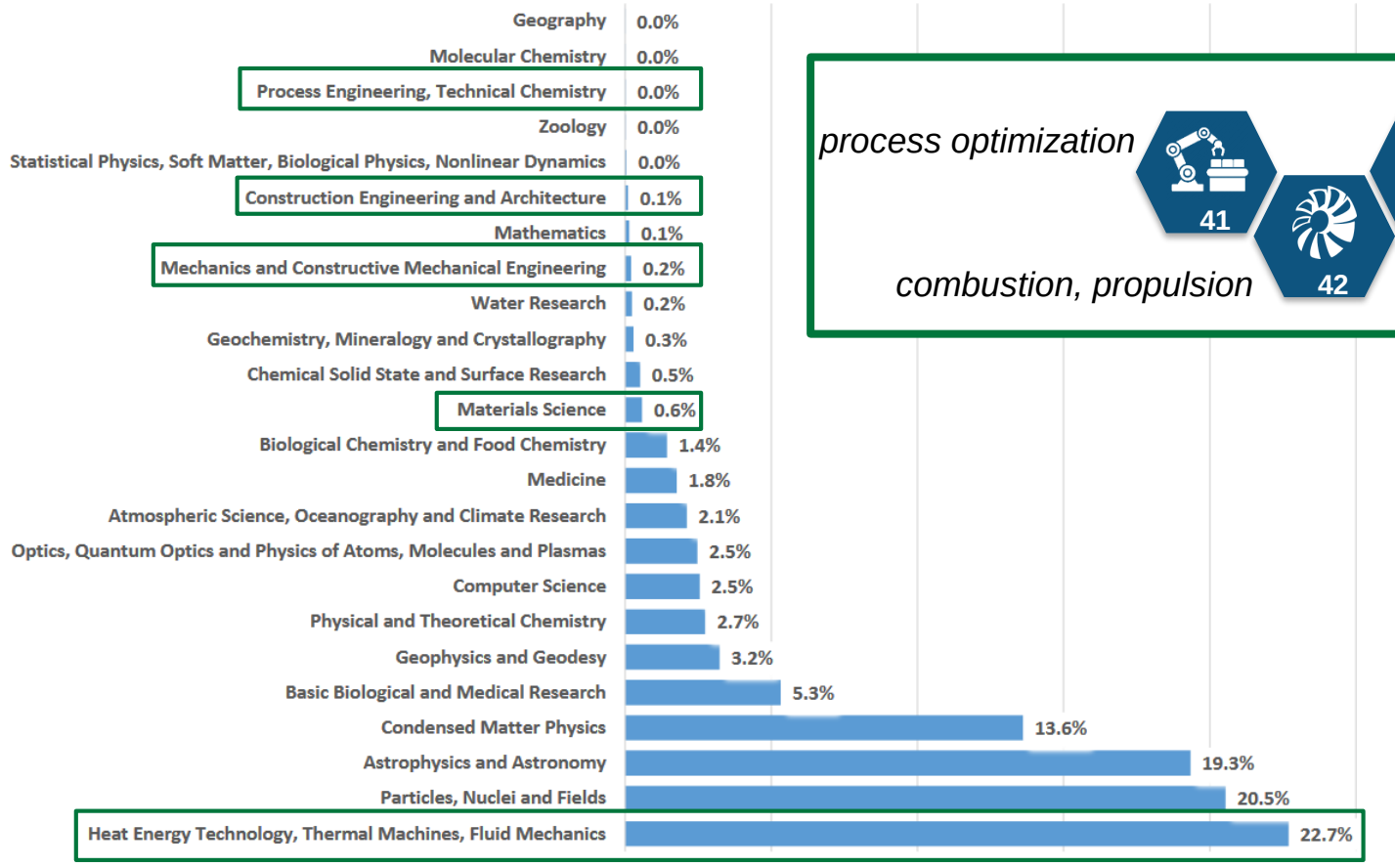
- ➔ *Enable **exchange** of **huge** high-quality **datasets**.*
- ➔ *Provision of HPC-data to foster **wide-spread usage**.*
- ➔ *Drive NFDI-wide **new methodologies** for data sharing*



DORIS's patron is Christian Stemmer

High Performance Measurement and Computing (HPMC)

Usage Statistics for SuperMUC-NG



https://doku.lrz.de/display/PUBLIC/Usage+Statistics+for+SuperMUC-NG?preview=/43320861/99680687/Statistics-SNG_2021.pdf

HPMC Research Data

What are HPMC Research Data

Research data are data that are created during a research process or are the result of it

High Performance Measurement

- Measurement data
- **Metadata** (hardware, method etc.)

Analysis and processing of measurement data using HPC

High Performance Computing

- Script / code (?)
- Input file, output file, log file
- Raw data
- Processed data
- **Metadata** (software, hardware, method etc.)
- Data for **secondary research** (e.g. energy consumption or temperature in HPC)

HPMC Research Data

Characteristics

- Data are created and stored in personalized accounts directly at HPC centres → no indexing by repositories or search engines
- Special hard- & software required for creating, reading or processing data
- Size: terabyte to petabyte → data is not mobile
- “Data” consists of various components (code, input file, raw data, metadata etc.)
- No established terminology or metadata scheme
- Little best-practice or showcases for research data management

Implementation of FAIR data principles

Findable: storage in personalized accounts, little metadata

Accessible: no access for third parties, insufficient transfer tools

Interoperable: depending on formats and enriched metadata

Reusable: computing time at HPC centres required or virtualization (e.g. container)

TUM Metadata Crawler: General purpose

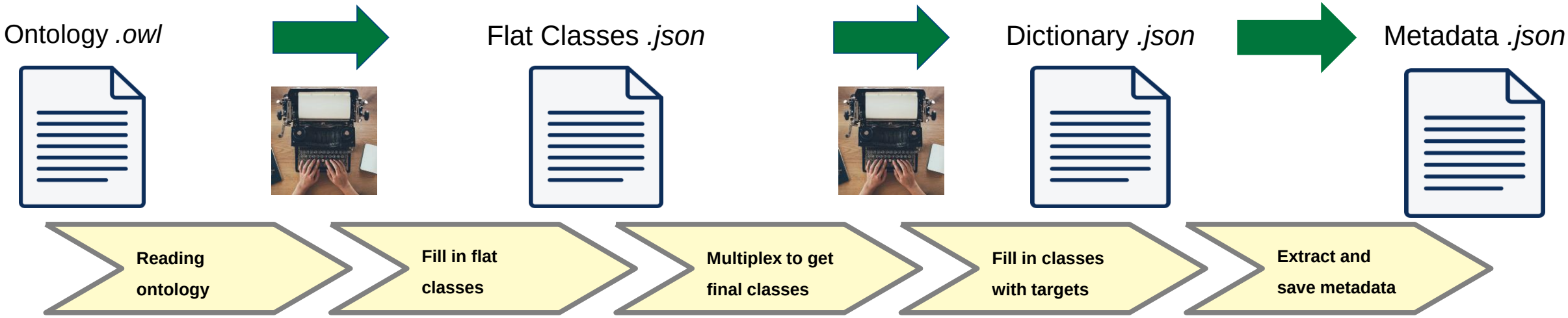
- To improve the FAIRness (Findable, Accessible, Interoperable, Reusable) of their data, researchers should properly manage the data they produce during the various steps of the research process
- A step in this direction is classifying produced data by producing metadata (i.e. data about data) to be attached each generated dataset, be it raw or post-processed
- If done manually, this job becomes easily burdensome and time-consuming with the increase of the produced data
- To relief scientists from this pain, a specific tool (for the moment called “Crawler”) is being developed at TUM-AER with the purpose of automated extraction of metadata from selected data files
- Currently, the Crawler is in a quite advanced phase and almost ready for release

TUM Metadata Crawler: Working principles

gitlab.lrz.de/nfdi4ing/crawler

GitLab

- Python-based application already available via GitLab
- Reads ontologies and helps in extracting metadata from selected data files
- With limited user input, metadata files can be generated in an automated way
- The key-players of the application are Ontologies, Flat Classes, Dictionaries and Metadata. The crawler is executed in 5 steps



TUM Metadata Crawler: Example – Pizza ontology

- The pizza ontology is a formal conceptualization of knowledge related to the pizza domain
- It includes classes such as: Pizza (ex. Margherita, Quattro formaggi...), Topping (Tomato, Mozzarella...), Base (Crispy, Gluten free…) and others
- https://protege.stanford.edu/ontologies/pizza/pizza.owl
- In this example, metadata are extracted from plain text files, but the crawler can also search hdf5 files
- The example is available in the gitlab repository

TUM Metadata Crawler: Example – Pizza ontology

- Situation: Three co-workers have the habit to eat pizza for lunch every day (Monday to Saturday) at the pizzeria around the corner. The menus of the pizzeria change randomly every day, but they always include: 3 pizzas, 3 red and 2 special (1 vegetarian) pizzas. The three habitual co-workers always go for the first white and the two special pizzas.
- Objective: from the menus (.txt) of a week, retrieve name (“main topping”) and price of the pizzas eaten by each person

Monday menu

```
******-- White Pizzas --******
White First choice: Monterosa
White First choice toppings: Speck, scamorza
White First choice price: 7.5
**----**
White Second choice: Ruchetta
White Second choice toppings: Mozzarella, rucola, pachini, grana
White Second choice price: 6.5
**----**
White Third choice: Focaccia
White Third choice toppings: Pizza bianca al sale
White Third choice price: 4.5

*******-- Red Pizzas --*******
Red First choice: Margherita
Red First choice toppings: Pomodoro, mozzarella
Red First choice price: 5.5
**----**
Red Second choice: Villa Borghese
Red Second choice toppings: Pomodoro, mozzarella, verdure grigliate
Red Second choice price: 7
**----**
Red Third choice: Perugina
Red Third choice toppings: Pomodoro, mozzarella, salsiccia
Red Third choice price: 6.5

*****-- Special Pizzas --*****
Special First choice: Carbonara
Special First choice toppings: Mozzarella, guanciale, uovo, pecorino romano, pepe nero
Special First choice price: 7.5
**----**
Special Second choice: Vegana
Special Second choice toppings: Pomodoro, funghi porcini, carciofini, olive nere
Special Second choice price: 7
```

Pizza example – Steps 1 and 2

- **Step 1**: Read the ontology → Get the json file with empty flat classes
- **Step 2**: Manually fill in the flat classes

Ontology *.owl*

```
…
"Vegetarian Pizza": {
        "__count__": 1,
        "__is_subclass_of__":
[Pizza],
        "__restrictions__":
["…"],
        "Price": [1],
        "Has_main_topping": [1],
},
…
```

```
…
"Special Pizza": {
        "__count__": 2,
        "__is_subclass_of__":
[Pizza],
        "__restrictions__": ["…"],
        "Price": [1,1],
        "Has_main_topping": [1,1],
},
…
```

Pizza example – Steps 3 and 4

- **Step 3**: Use the multiplexer to expand the classes count as needed
- **Step 4**: Manually fill in the extended dictionary

```
…
"Special Pizza_1": {
            "__restrictions__": ["…"],
            "Price": {
                        "path": [],
                        "type": [],
                        "pattern": [],
                        "postprocessor": [ {
                                    "type": "",
                                    "args": ""
                                    }]
            }
            "Has_main_topping": {
                        "path": [],
                        "type": [],
                        "pattern": [],
                        "postprocessor": [ {
                                    "type": "",
                                    "args": ""
                                    }]
            }
},
"Special Pizza_2": {
            …
}
…
```

```
…
"Special Pizza_1": {
            "__restrictions__": "…",
            "Price": {
                        "path":
"01_Example_Pizza_NFDIConf2022/Menus/RandomMenu_1.txt",
                        "type": "regex",
                        "pattern": "Special First choice price:\\s(.*)\\",
                        "postprocessor": {
                                    "type": "",
                                    "args": ""
                                    }
            }
            "Has_main_topping": {
                        "path":
"01_Example_Pizza_NFDIConf2022/Menus/RandomMenu_1.txt",
                        "type": "regex",
                        "pattern": "Special First choice price:\\s(.*)\\",
                        "postprocessor": {
                                    "type": "",
                                    "args": ""
                                    }
            }
},
"Special Pizza_2": {
            …
```

Pizza example – Step 5

Step 5: Metadata extraction from target files to new .json or .yaml file

Monday

```
{
        "Special Pizza_1": {
                "Price": "7.5",
                "Has_main_topping": "Carbonara"
        },
        "Special Pizza_2": {
                "Price": "7",
                "Has_main_topping": "Vegana"
        },
        "White Pizza": {
                "Price": "7.5",
                "Has_main_topping": "Monterosa"
        }
}
```

Pizza example – Automate steps 4 and 5

- Now, only steps 4 and 5 have to be re-run to extract similar data from different target files
- This can be easily automated (ex. via simple shell script)

Tuesday


```
{
    "Special Pizza_1": {
        "Price": "8",
        "Has_main_topping": "Romoletta"
    },
    "Special Pizza_2": {
        "Price": "8",
        "Has_main_topping": "Pariolina"
    },
    "White Pizza": {
        "Price": "4.5",
        "Has_main_topping": "Focaccia"
    }
}
```


Wednesday


```
{
    "Special Pizza_1": {
        "Price": "7.5",
        "Has_main_topping": "Carbonara"
    },
    "Special Pizza_2": {
        "Price": "8",
        "Has_main_topping": "Pariolina"
    },
    "White Pizza": {
        "Price": "7.5",
        "Has_main_topping": "Quattro Formaggi"
    }
}
```


Thursday


```
{
    "Special Pizza_1": {
        "Price": "7.5",
        "Has_main_topping": "Carbonara"
    },
    "Special Pizza_2": {
        "Price": "8",
        "Has_main_topping": "Pariolina"
    },
    "White Pizza": {
        "Price": "7.5",
        "Has_main_topping": "Quattro Formaggi"
    }
}
```


Friday


```
{
    "Special Pizza_1": {
        "Price": "8",
        "Has_main_topping": "Romoletta"
    },
    "Special Pizza_2": {
        "Price": "8",
        "Has_main_topping": "Pariolina"
    },
    "White Pizza": {
        "Price": "4.5",
        "Has_main_topping": "Focaccia"
    }
}
```


Saturday


```
{
    "Special Pizza_1": {
        "Price": "9",
        "Has_main_topping": "Salmonata"
    },
    "Special Pizza_2": {
        "Price": "8",
        "Has_main_topping": "Pariolina"
    },
    "White Pizza": {
        "Price": "7.5",
        "Has_main_topping": "Quattro Formaggi"
    }
}
```

CFD example

- While it is possible to read out a subject-specific ontology to generate the first dictionary with flat classes, it is not necessary, as steps 3 and 4 are independent of the actual names of the classes
- This means that, with some further user input, the crawler can be already applied to real problems, like CFD workflows

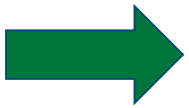
```
…
"Vegetarian Pizza": {
        "__count__": 1,
        "__is_subclass_of__": [Pizza],
        "__restrictions__": ["…"],
        "Price": [1],
        "Has_main_topping": [1]
},
…
```

```
{
        "Processing_Step": {
                "__count__": 1,
                "__is_subclass_of__":
["",],
                "__restrictions__": ["…"],
                "Name": [1],
                "Parameter": [5],
                "Has_numerical_value": [5]
        },
        "Tool": {
                "__count__": 2,
                "__is_subclass_of__":
["",],
                "__restrictions__": ["…"],
                "Type": [1],
                "Name": [5],
                "Version": [5]
        },
        "Method": {
                "__count__": 1,
                "__is_subclass_of__":
["",],
                "__restrictions__": ["…"],
                "Name": [1]
```

CFD example

- For example, it's possible to retrieve some physical parameters used in a simulation (as the freestream Mach number) from the input file of a CFD code or numerical parameters from the output file

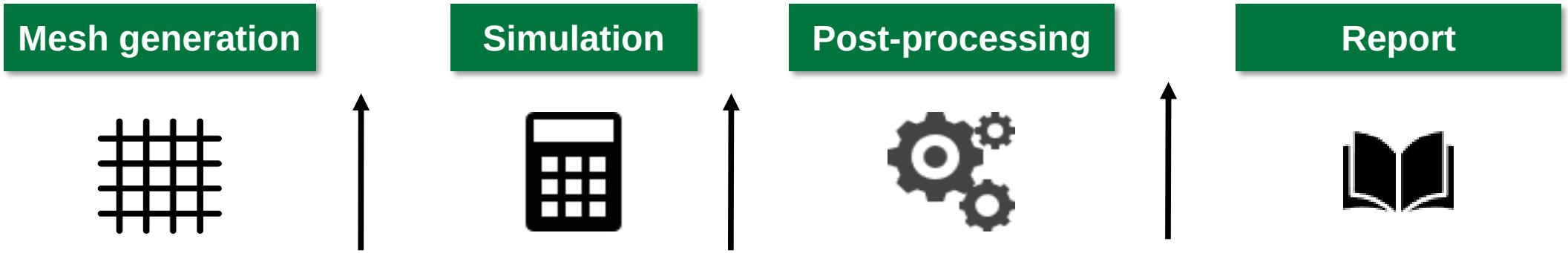
```
{
"Processing_Step": {
            "__restrictions__": "",
            "Name": {
                        "path": "whatever",
                        "type": "os",
                        "pattern": "echo 'Running Simulation'",
                        "postprocessor": {
                                    "type": "",
                                    "args": ""
                                    }
            },
            "Parameter_1": {
                        "path": "whatever",
                        "type": "os",
                        "pattern": "echo 'Freestream Mach number'",
                        "postprocessor": {
                                    "type": "",
                                    "args": ""
                                    }
            },
            "Has_numerical_value_1": {
                        "path": "02_CFDEx_NSMB_NFDIConf2022/example_NSMB_input.dat",
                        "type": "regex",
                        "pattern": "Mach :\\s(.*)\\",
                        "postprocessor": {
                                    "type": "",
                                    "args": ""
                                    }
```



```
{
    "Processing_Step": {
        "Name": "Running Simulation\n",
        "Parameter_1": "Freestream Mach number\n",
        "Parameter_2": "Freestream pressure\n",
        "Parameter_3": "Freestream temperature\n",
        "Parameter_4": "Freestream unit Reynolds number\n",
        "Parameter_5": "Start time\n",
        "Has_numerical_value_1": "9.1",
        "Has_numerical_value_2": "730",
        "Has_numerical_value_3": "160",
        "Has_numerical_value_4": "3.22E6",
        "Has_numerical_value_5": "10:13:31"
    },
    "Tool_1": {
        "Type": "Hardware\n",
        "Name": "lrz-coolmuc2-linux-cluster-2022\n"
    },
    "Tool_2": {
        "Type": "Software\n",
        "Name": "NSMB\n",
        "Version": " 6.09.21    Date:  28 - January - 2021     "
    },
    "Method": {
        "Name": "LU-SGS"
    }
}
```

TUM Metadata Crawler – Application in HPC workflow

- Depending on the application, the crawler can be used at different steps within the workflow of CFD (or similar) applications
- Wherever data files are created, the crawler can be used to extract relevant metadata



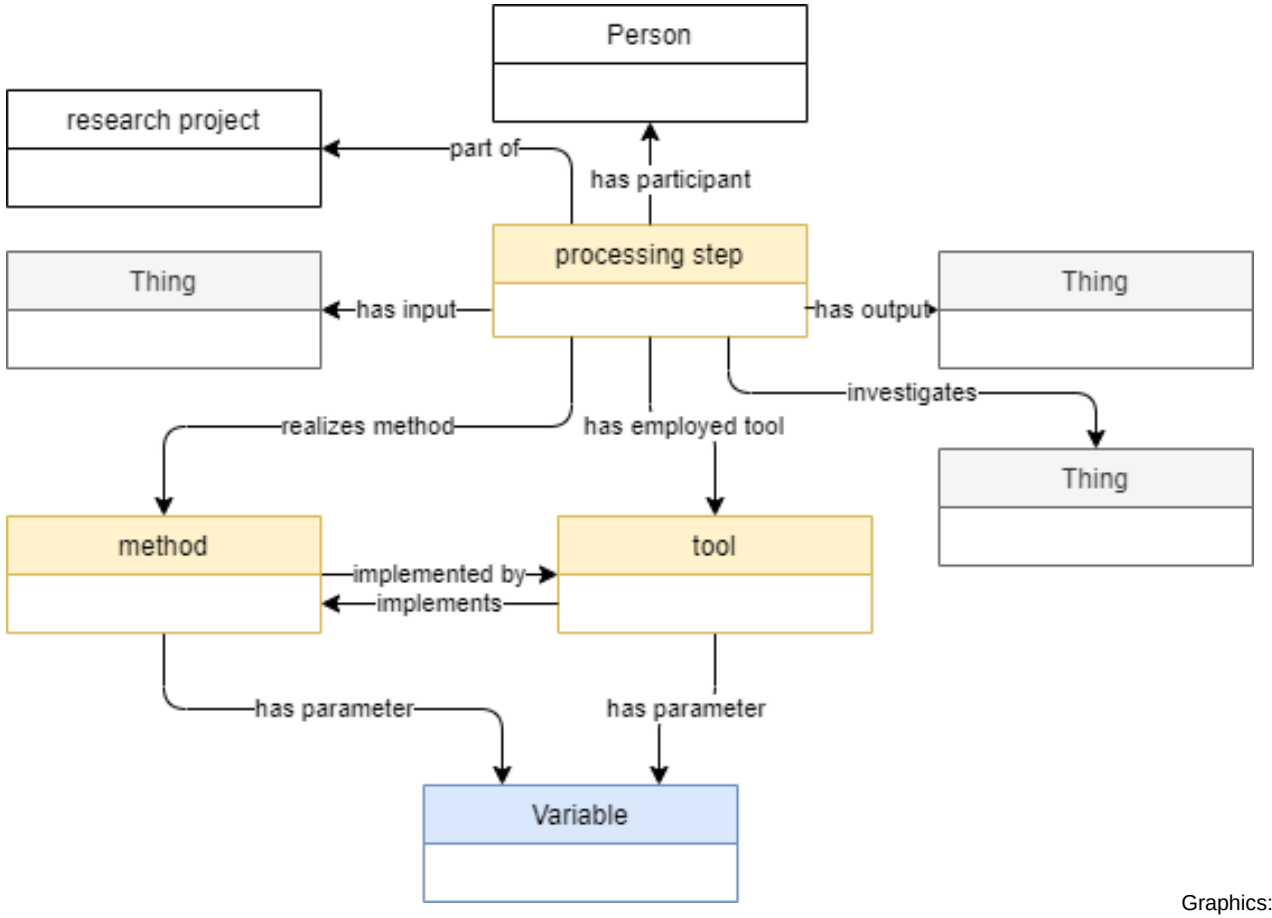
TUM Metadata Crawler – Next steps

- Although HDF5 can be already searched for data, some more work is needed in that area
- Increase the number of supported formats
- Write documentation
- Implement complete Metadata4Ing ontology
- Implement/improve post-processing routines on the extracted data
- ...

HPMC workflows in Metadata4Ing

https://git.rwth-aachen.de/nfdi4ing/metadata4ing/metadata4ing

GitLab



13:00 - 13:30 ID: 817

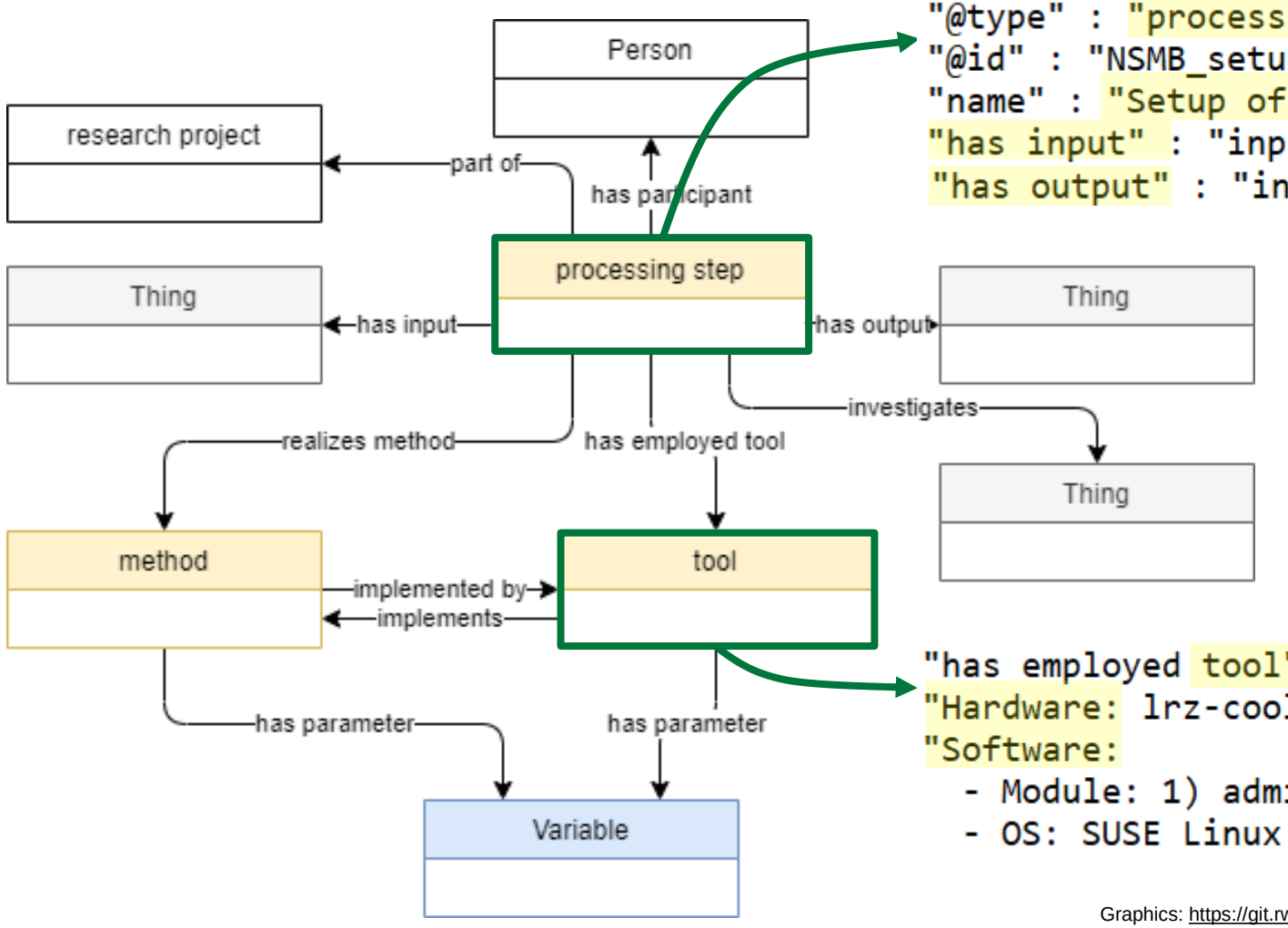
» Metadata4Ing: An ontology for describing the generation and provenance of research data within a scientific activity ↗

Susanne Arndt
(TIB – Leibniz Information Centre for Science and Technology and University Library, Hannover, Germany)

Metadata4ing-tutorial

Graphics: https://git.rwth-aachen.de/nfdi4ing/metadata4ing/metadata4ing (modified)

HPMC workflows in Metadata4Ing

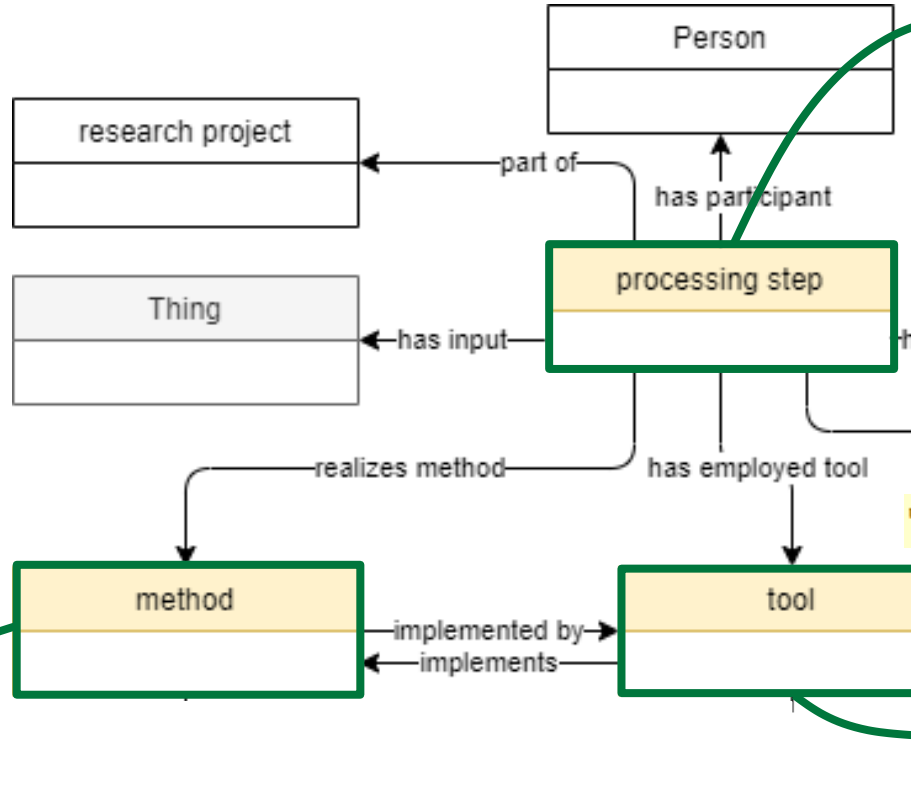



```
"@type" : "processing step",
"@id" : "NSMB_setup",
"name" : "Setup of simulation parameters",
"has input" : "input.dat",
"has output" : "input.dat",  // Comment: modifi
```

```
"has employed tool" : [
"Hardware: lrz-coolmuc2-linux-cluster-2022",
"Software:
  - Module: 1) admin/1.0   2) tempdir/1.0   3)
  - OS: SUSE Linux Enterprise Server 15 SP1, CP
```

Graphics: https://git.rwth-aachen.de/nfdi4ing/metadata4ing/metadata4ing (modified)

HPMC workflows in Metadata4Ing




```
"@type" : "processing step",
"@id" : "NSMB_running-simulation",
"name" : "Running Simulation",
"has input" : [
  "input.dat"
  "RG_flat_plate.db"
```

```
"investigates": "Boundary layer on a flat plate",
```

```
"has employed tool":
["Hardware: lrz-coolmuc2-linux-cluster-2022",
"Software:
  - Module: 1) admin/1.0   2) tempdir/1.0   3)
  - OS: SUSE Linux Enterprise Server 15 SP1, CP
  - Executable: NSMB_6.09.18.mpi // Comment: di
```

```
"realizes method": "PDE solver (Navier-Stokes, compressible)",
```

Graphics: https://git.rwth-aachen.de/nfdi4ing/metadata4ing/metadata4ing (modified)

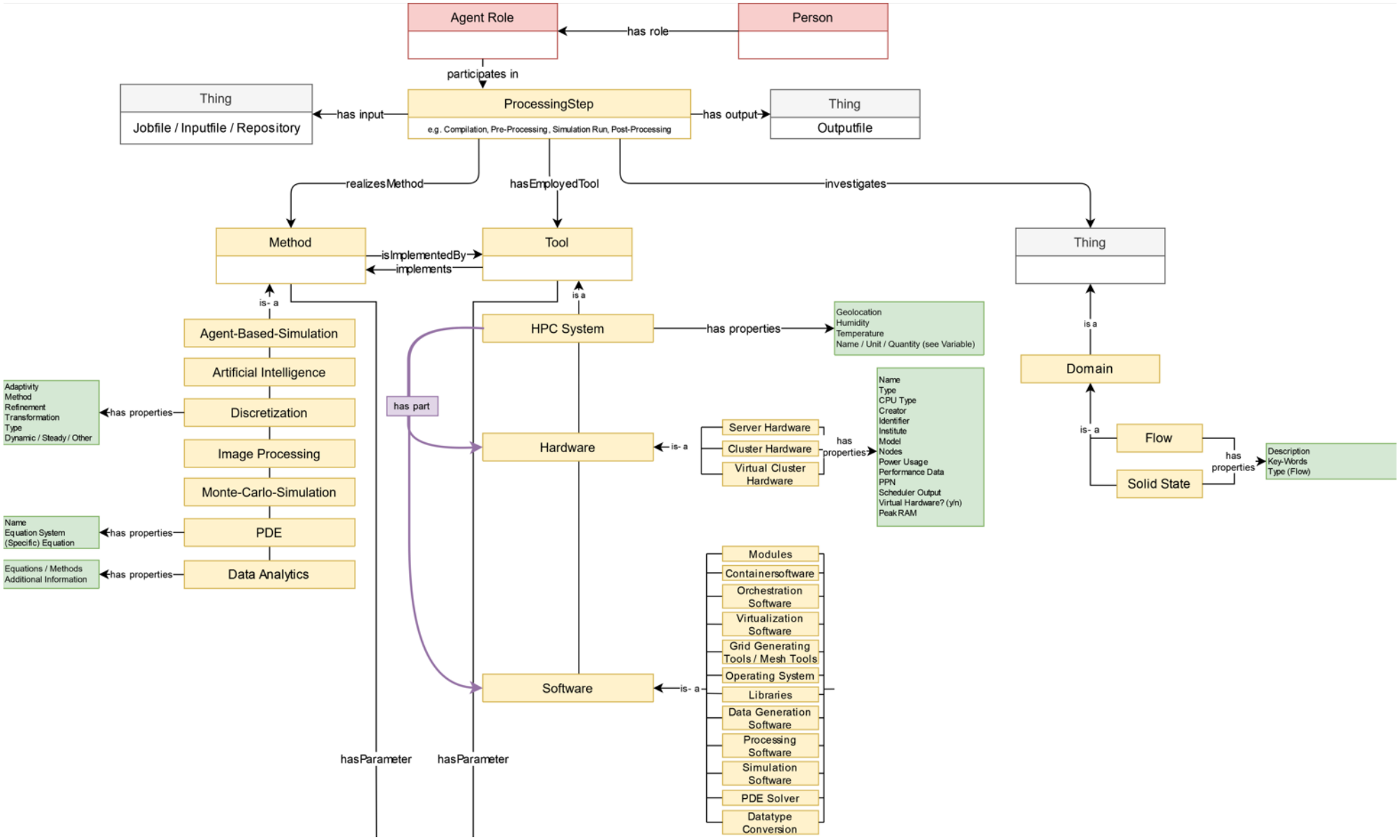
HPMC workflows in Metadata4Ing

HPMC extension / domain-ontology

- Community based, consistent terminology for HPMC
- https://git.rwth-aachen.de/nfdi4ing/metadata4ing/metadata4ing/-/tree/DORIS_HPMC

- Set classes and properties
  - **Tool**
    → HPC system (“*has part*:” hardware & software)
  - **Method**
    → PDE, Monte-Carlo-Simulation, Image processing etc.
  - **Processing Step**
    → Compilation, Pre-Processing, Simulation run, Post-Processing etc.
  - **Domain**
    → Flow, Solid state

  - optional: detailed metadata, e.g. energy consumption, used nodes, temperature in cluster etc.
    → useful for secondary research


Agent Role
has role
Person
participates in
Thing
Jobfile / Inputfile / Repository
has input
ProcessingStep
e.g. Compilation, Pre-Processing, Simulation Run, Post-Processing
has output
Thing
Outputfile
realizesMethod
hasEmployedTool
investigates
Method
isImplementedBy
implements
Tool
Thing
is a
is- a
Agent-Based-Simulation
Artificial Intelligence
Discretization
Image Processing
Monte-Carlo-Simulation
PDE
Data Analytics
Adaptivity
Method
Refinement
Transformation
Type
Dynamic / Steady / Other
has properties
Name
Equation System
(Specific) Equation
has properties
Equations / Methods
Additional Information
has properties
has part
HPC System
has properties
Geolocation
Humidity
Temperature
Name / Unit / Quantity (see Variable)
Hardware
is- a
Server Hardware
Cluster Hardware
Virtual Cluster Hardware
has properties
Name
Type
CPU Type
Creator
Identifier
Institute
Model
Nodes
Power Usage
Performance Data
PPN
Scheduler Output
Virtual Hardware? (y/n)
Peak RAM
Software
is- a
Modules
Containersoftware
Orchestration Software
Virtualization Software
Grid Generating Tools / Mesh Tools
Operating System
Libraries
Data Generation Software
Processing Software
Simulation Software
PDE Solver
Datatype Conversion
hasParameter
hasParameter
Domain
is a
is- a
Flow
Solid State
has properties
Description
Key-Words
Type (Flow)

HPMC workflows in Metadata4Ing

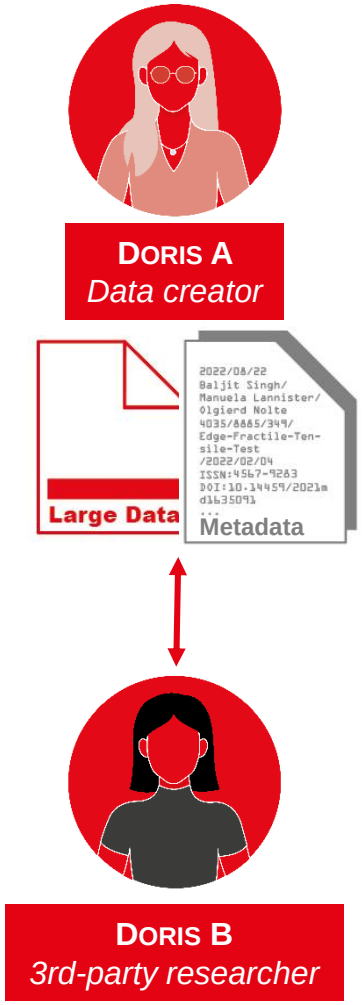
HPMC extension / domain-ontology – Why?

- Provide a comprehensive, expandable metadata schema for (engineering) research data generated on HPMC systems
- Establish consistent terminology for HPMC in engineering

Further steps

- Evaluate schema along with pilot users
- Currently: CFD-approach >> other perspectives
- Work out details, e.g. discretization, data quality
- Improved mapping of software (tool)
- Merge with LRZ/DataCite schema

Storage and Publication of Metadata



Findability & Accessibility for Re-Use of (Meta-)Data

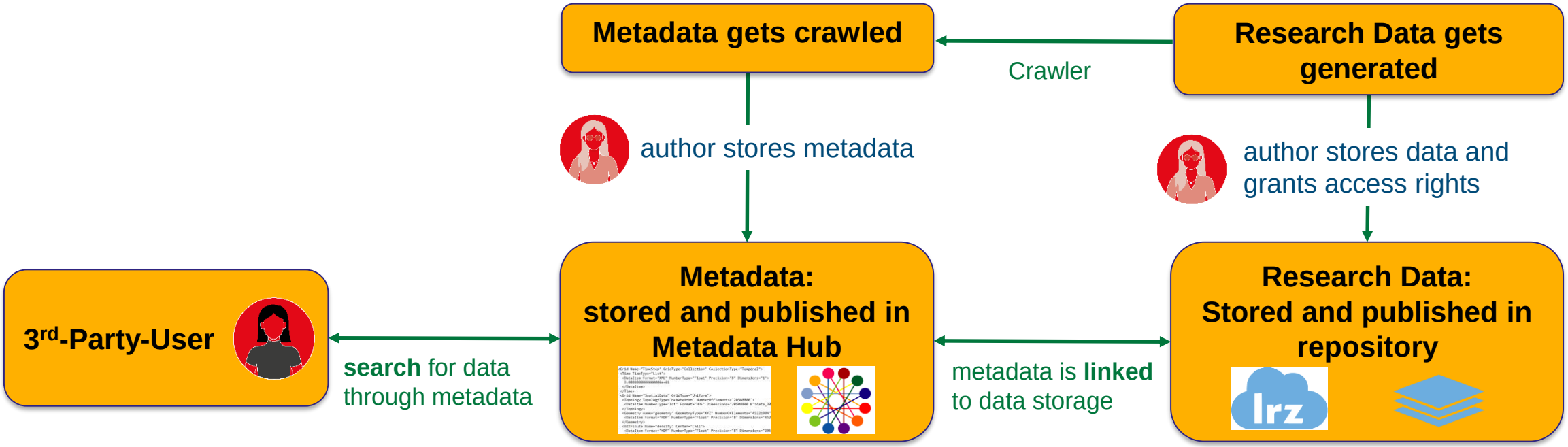
- Storage & publication in institutional repository along with published research data
  - not indexable ☹

→ Pending: repository linking or storage in NFDI4Ing **MetadataHub**

https://nfdi4ing.de/base-services/s-3/
https://git.rwth-aachen.de/nfdi4ing/s-3/s-3-3/metadatahub

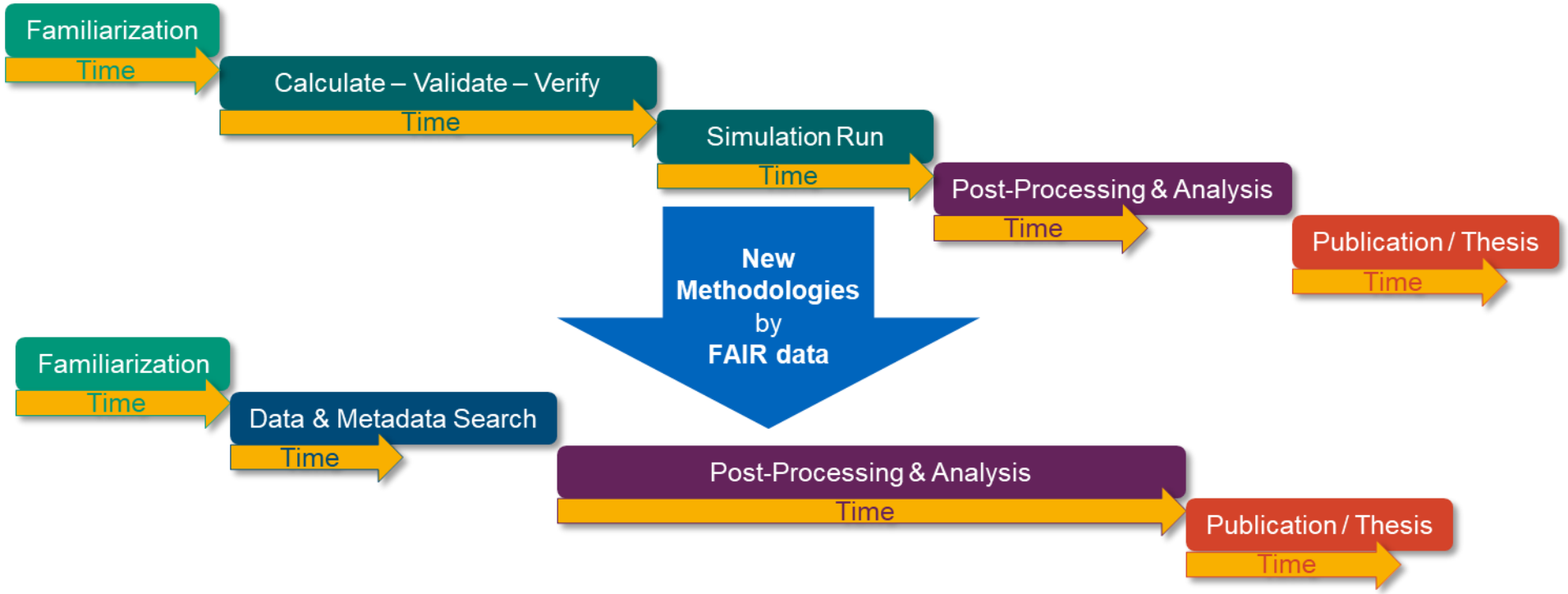
  - linked (raw) data, metadata, software, citations etc.
  - standardized metadata schemes
  - indexing of metadata sets, enabling **metadata-based search for research data**

Storage and Publication of Metadata



→ *(large) data is stored at HPC centre*
→ *metadata is stored in a indexable metadata hub*
→ *3rd party users can search, find and re-use data through metadata*

Usage of FAIR data from HPMC



New methodologies as alternative (not as replacement)

Further Information

Follow-Up

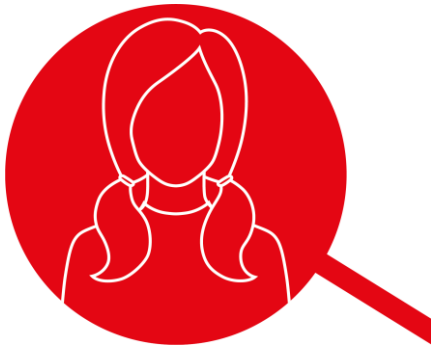
- DORIS **workshop on RDM in HPMC** (feat. JSC, HLRS, LRZ): 2023, 1st quarter

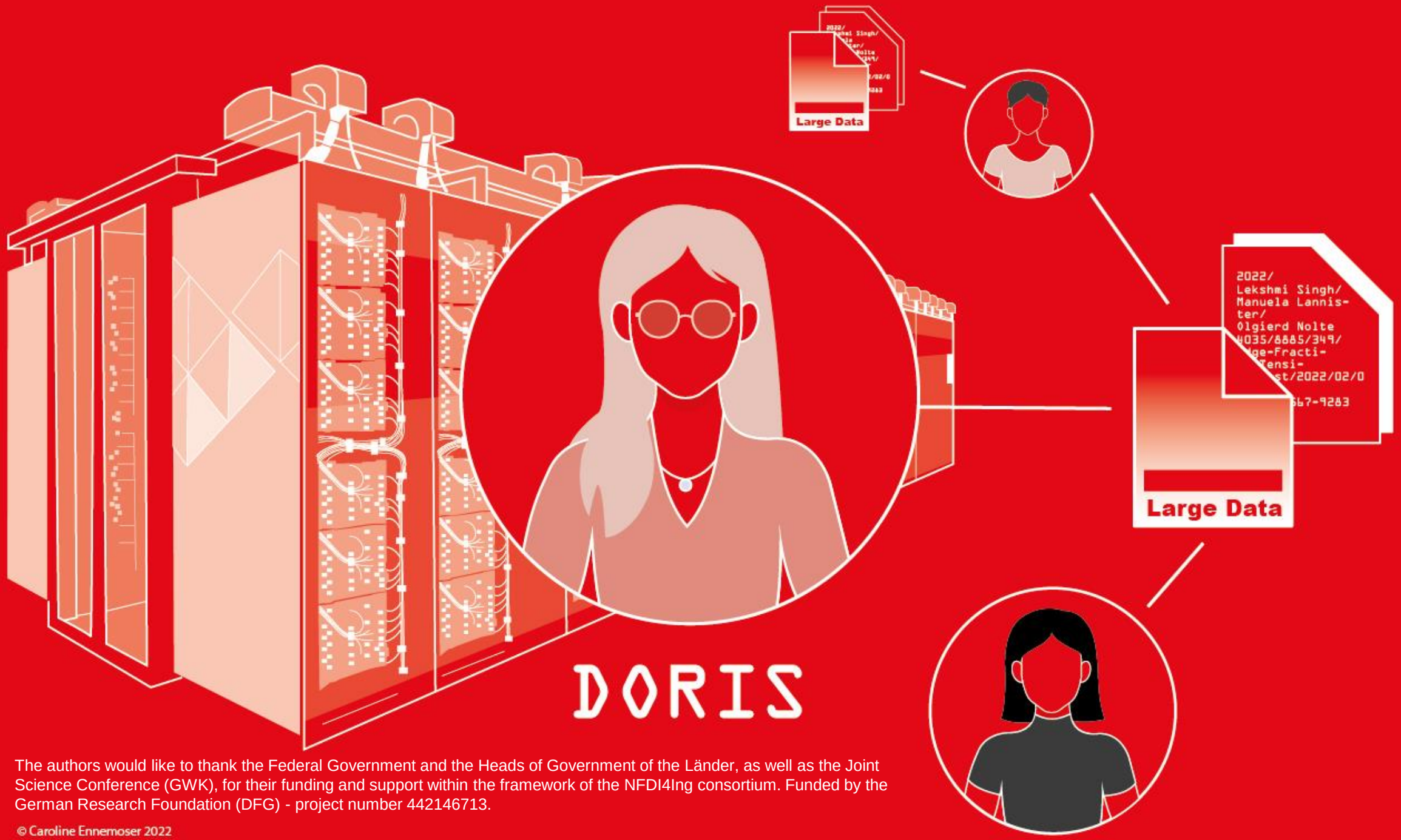
Downloads

- **Software**: https://gitlab.lrz.de/nfdi4ing
- **Metadata4Ing**: https://git.rwth-aachen.de/nfdi4ing/metadata4ing

Contact

- **Newsletter**: https://lists.tu-darmstadt.de/mailman/listinfo/nfdi4ing_taskarea_doris
- **Mail**: info-doris@nfdi4ing.de
- **Web**: https://nfdi4ing.de/archetypes/doris/





The authors would like to thank the Federal Government and the Heads of Government of the Länder, as well as the Joint Science Conference (GWK), for their funding and support within the framework of the NFDI4Ing consortium. Funded by the German Research Foundation (DFG) - project number 442146713.