VMAP USER FORUM 2025

Gefördert durch:

Bundesministerium für Wirtschaft und Klimaschutz

aufgrund eines Beschlusses des Deutschen Bundestages

VMAP AS A KEY ENABLER FOR THE SUSTAINABILITY & COMPONENT QUALITY ASSESSMENT OF COMPOSITE MANUFACTURING PROCESSES BASED ON SIMULATIVE APPROACHES

Andreas Schuster and Martin Rädel and Jean Lefèvre

Institute of Lightweight Systems, DLR

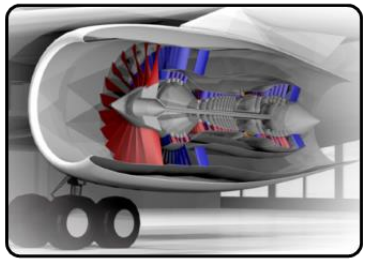
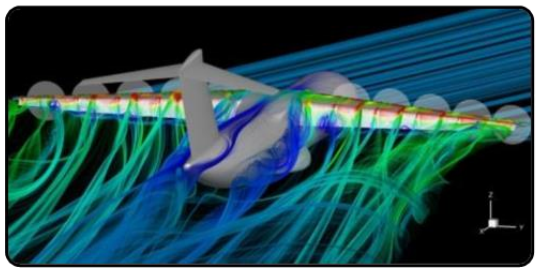
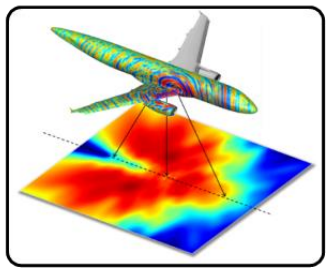
PredictECO

DLR

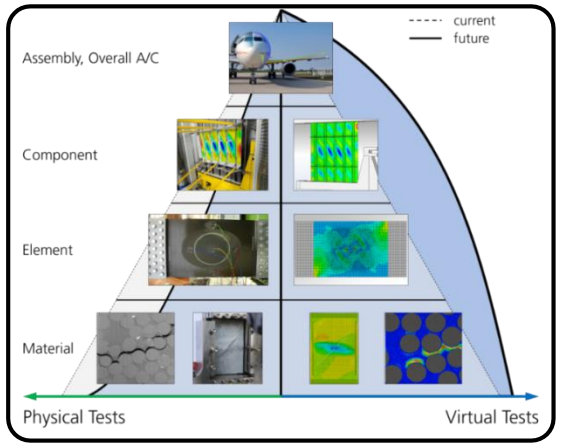
Digitalisation in Aeronautics & Virtual Product

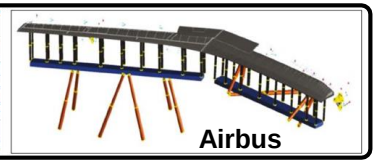
Digital Aircraft



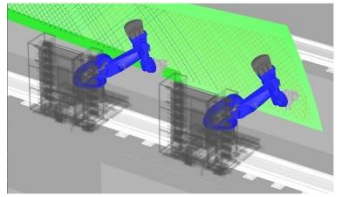
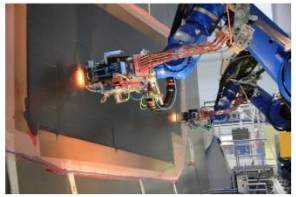





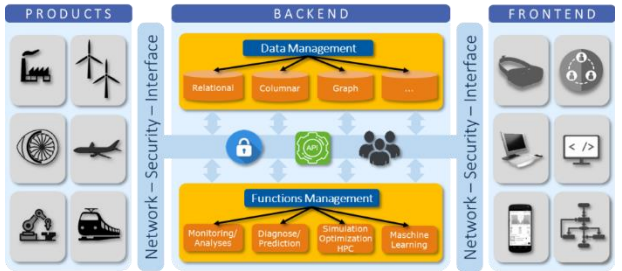
Virtual Certification



Digital Twin

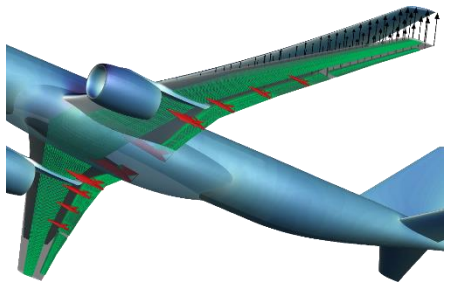



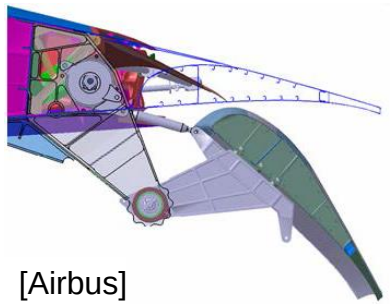


Further information:
www.dlr.de/vph

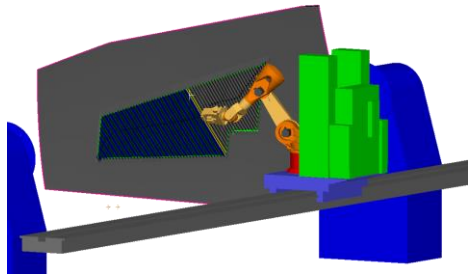
Digital End-to-End-Process

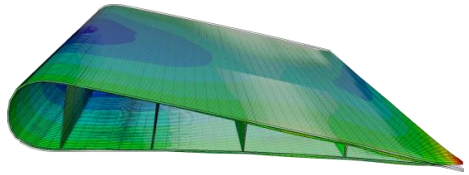
Further information:
www.dlr.de/vph

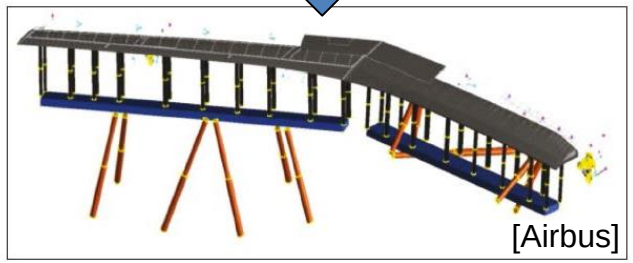




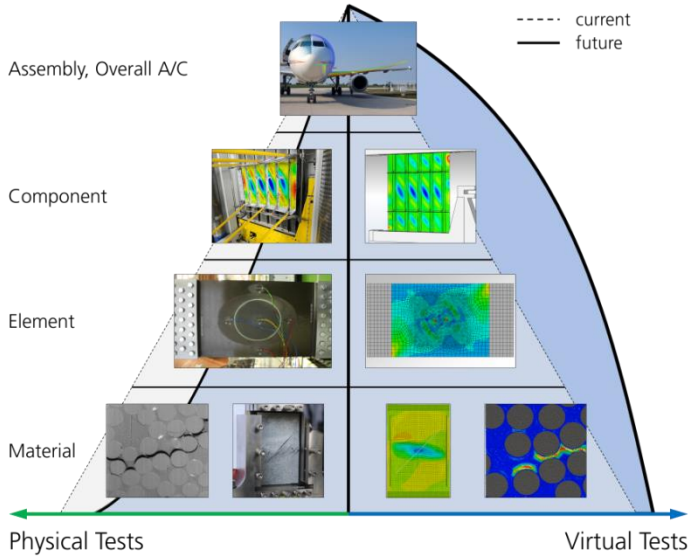
[Airbus]







[Airbus]

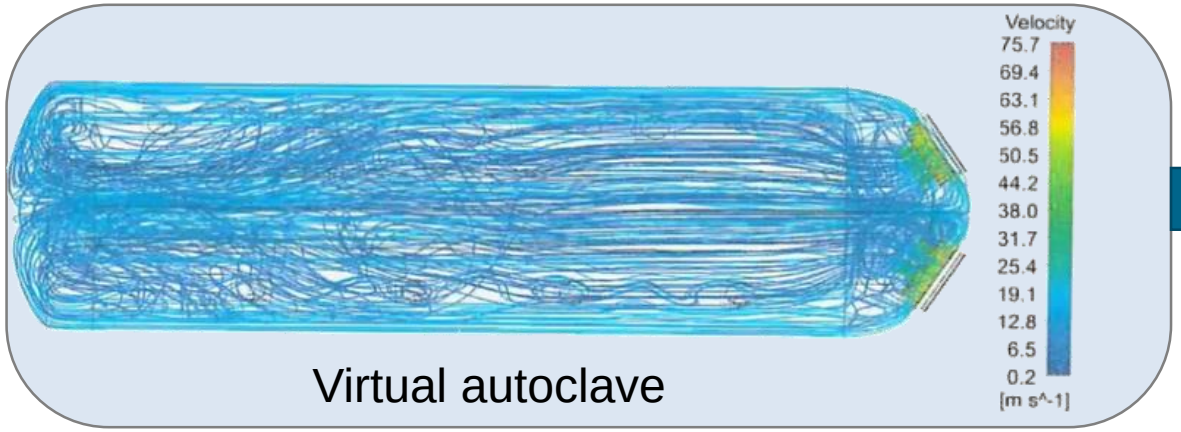


Virtual Design → Virtual Production → Virtual Testing → Virtual Certification

EASA European Aviation Safety Agency

→ Close cooperation w. industry, e.g. @ **Airbus:** FPO, flight physics, structures, systems, testing, DDMS

Manufacturing Simulation
Overview

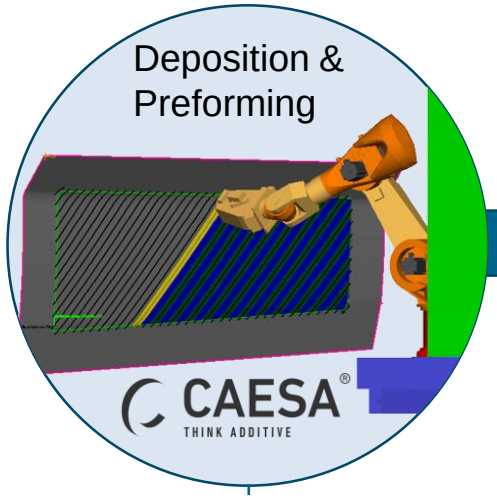


Virtual autoclave

Temperature
Heat transfer

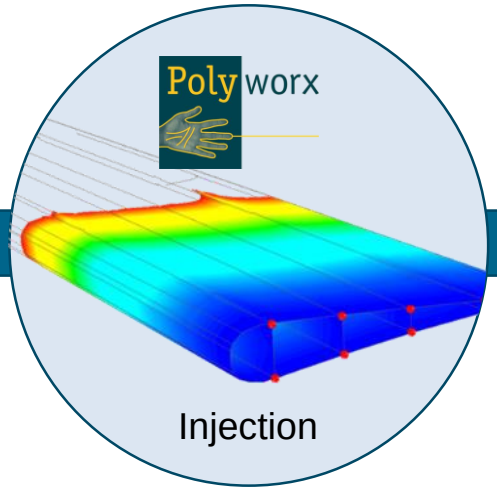
On specimen surface f(time):

Deposition &
Preforming



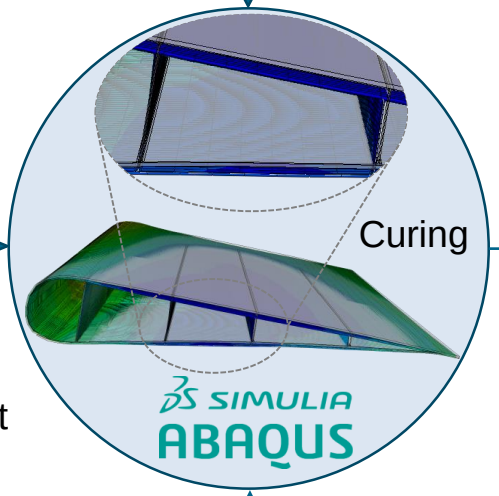
@ final state:

Gaps & overlaps



Injection

@ final state:
Fibre volume content
Degree of cure

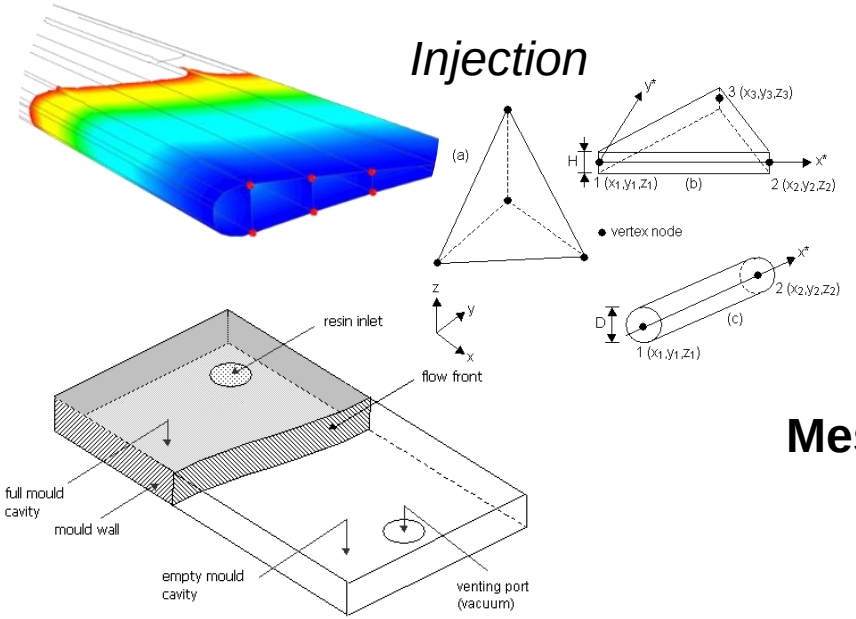


Curing

Requirements

Manufacturing Simulation
Overview

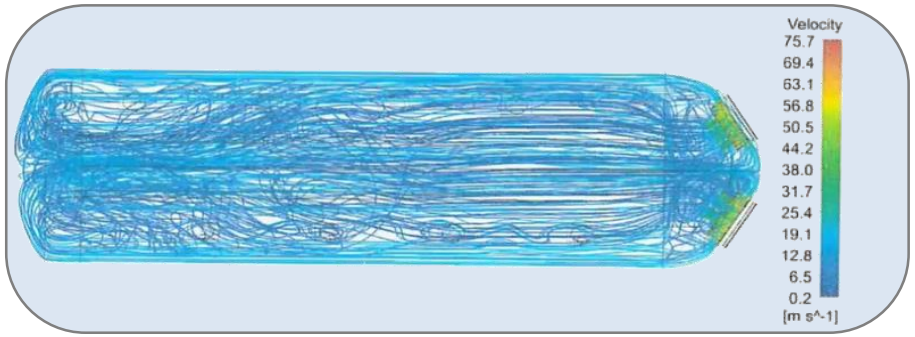
Injection



→ 2.5D based unstructured meshes

Mesh Interpolation

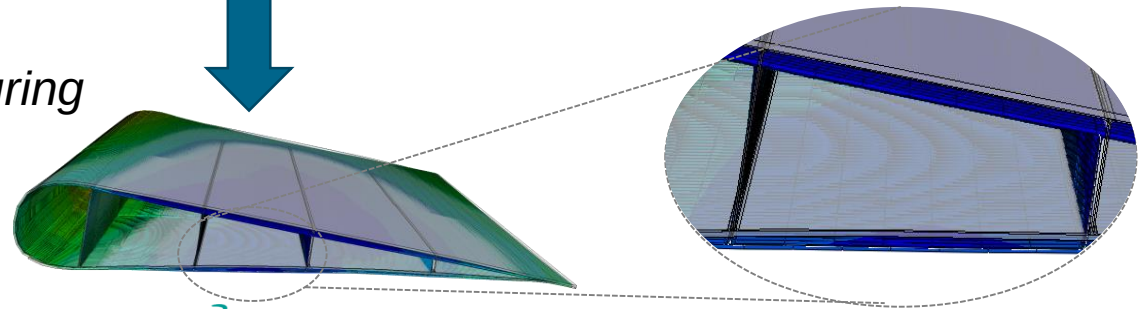
Virtual autoclave



→ point cloud based results

→ *Need for standardized data schemes and tools!*

Curing



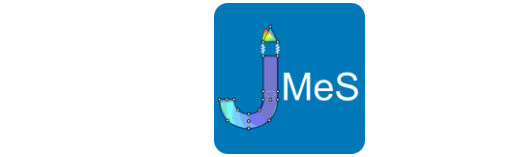
SIMULIA ABAQUS

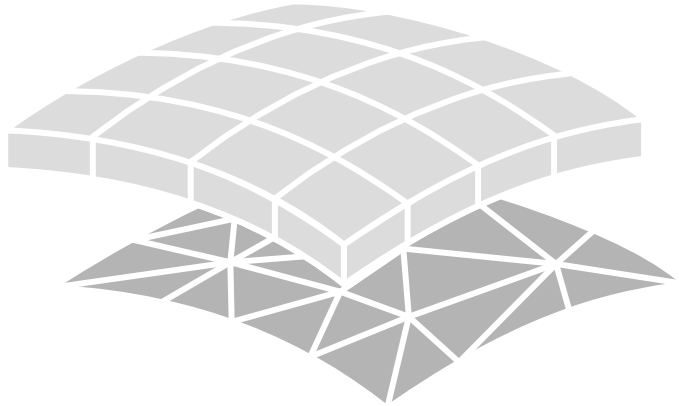
3D FE-Model

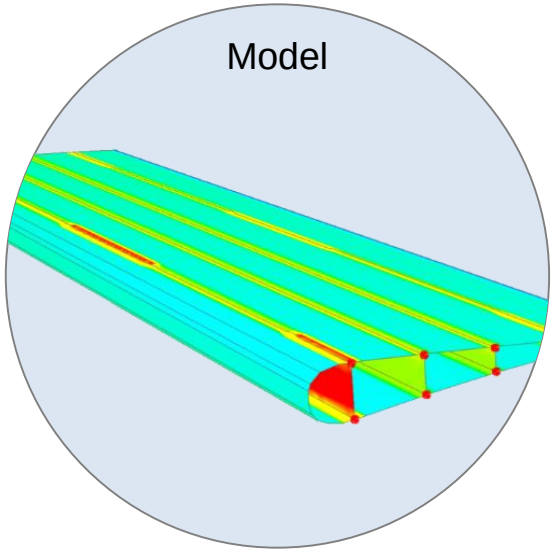
Data
Exchanges

Simulation

vampire

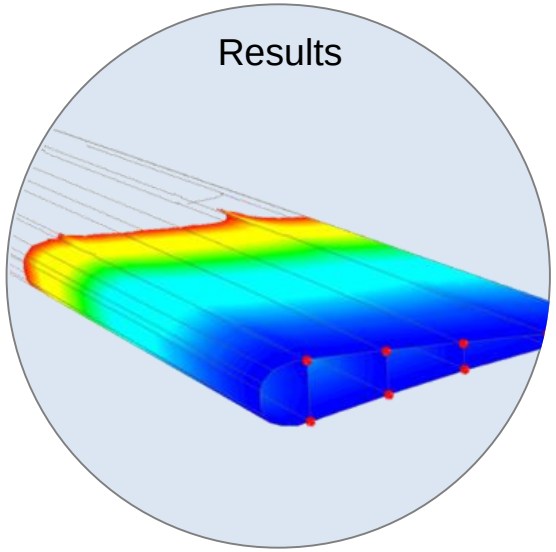




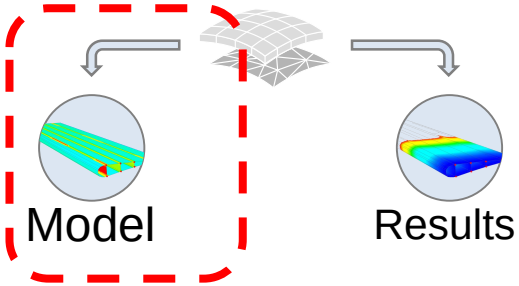


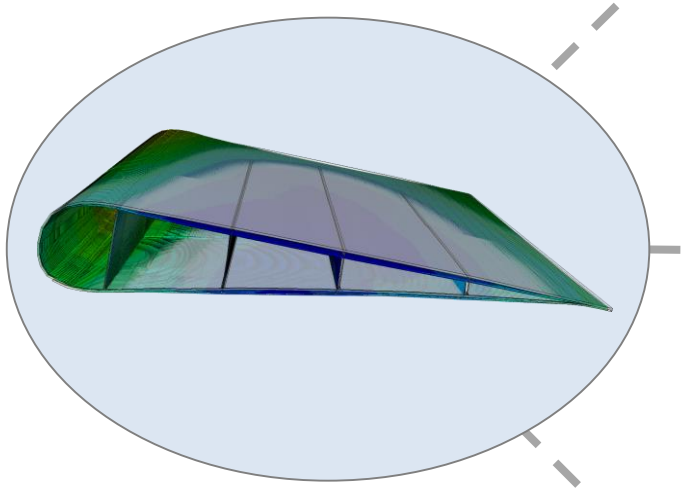
- Physical data
  - Temperature, …
- Mechanical data
  - Constituents, …
- CSM data
  - Discretization, …
  - Loads & BC

- Model data
- Part data
  - Nodes
  - Elements
  - Int.-Points
- Conservation data



Data
Standardization – Data types





Numerical implementation data

- Nodes
- Elements
- …

- Results

Connecting data

- Sections
- Loads & BC
- …

- Solution

Math, physics & mech. idealization data

- Materials
- Composites
- Crosssections
- …

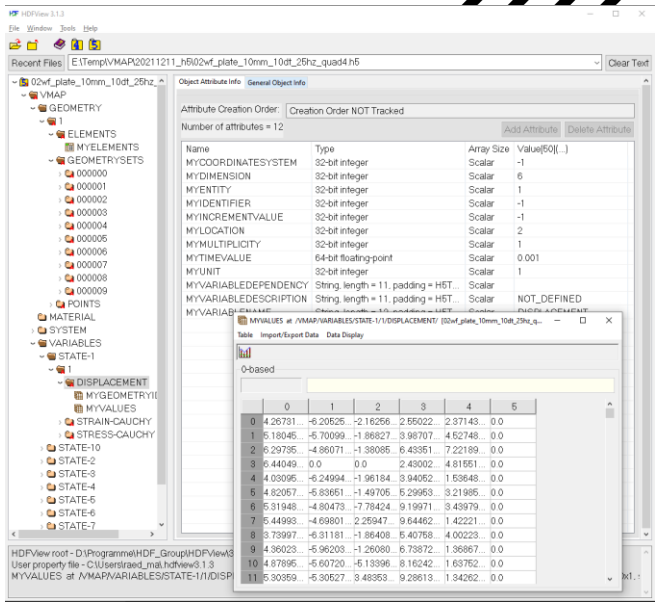
Heavy data

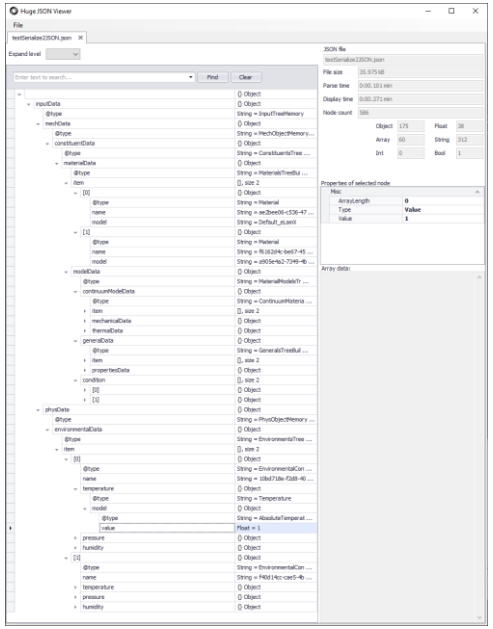
VMAP

References

Light data

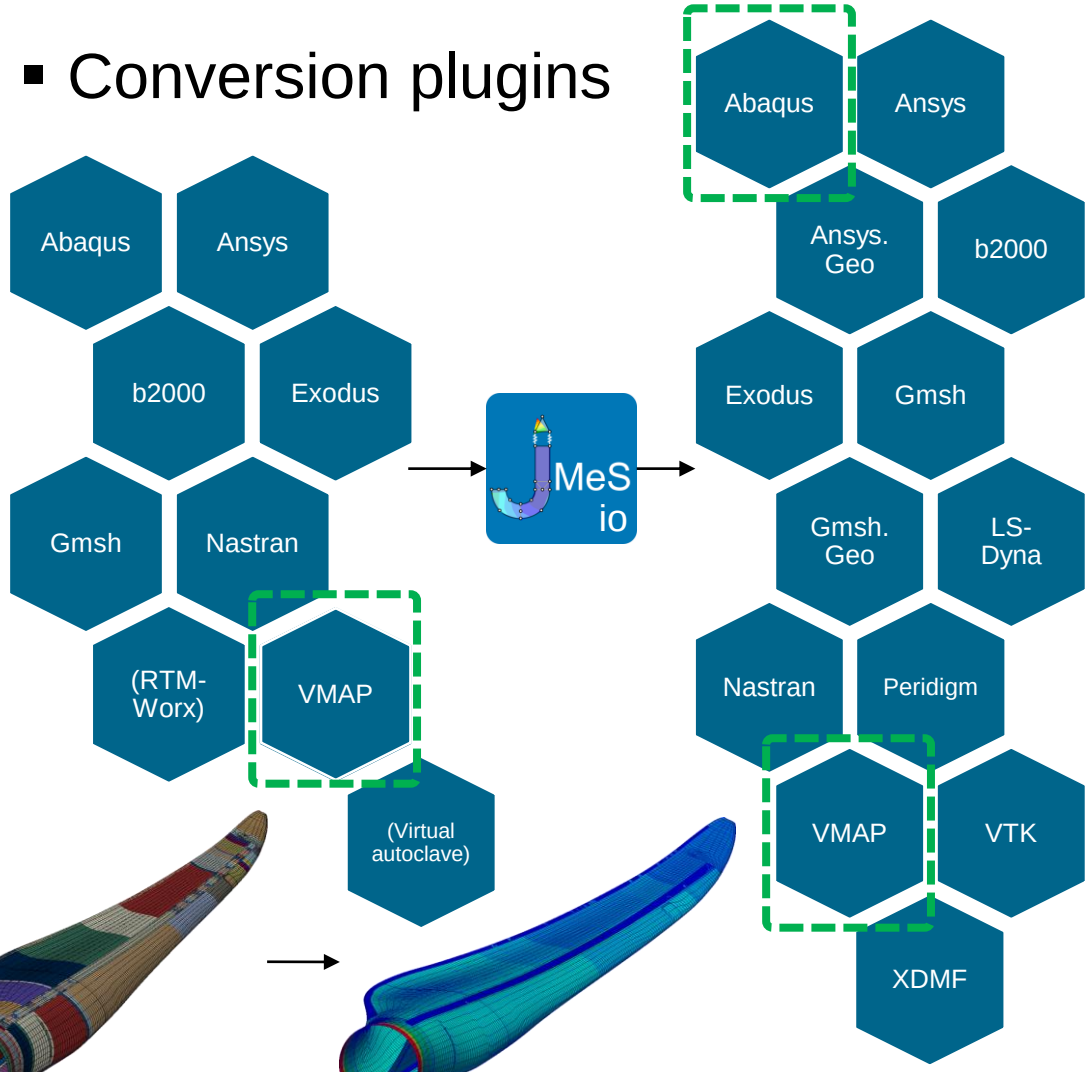
cosmo



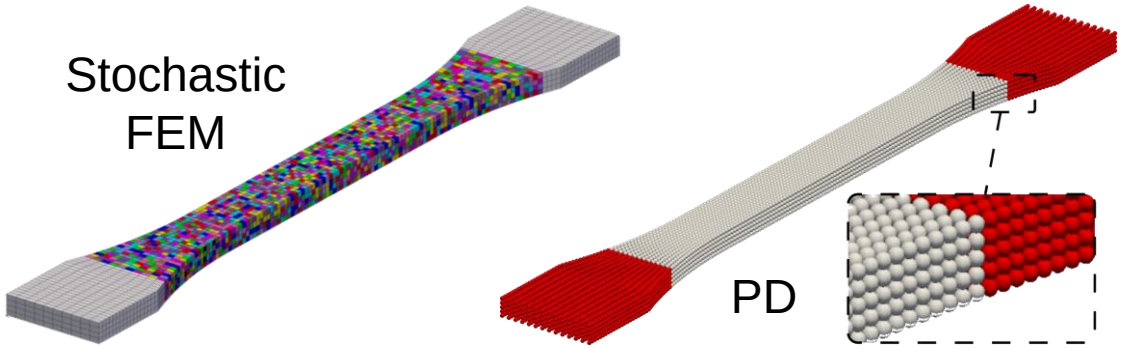


Implementation
Model – Functions

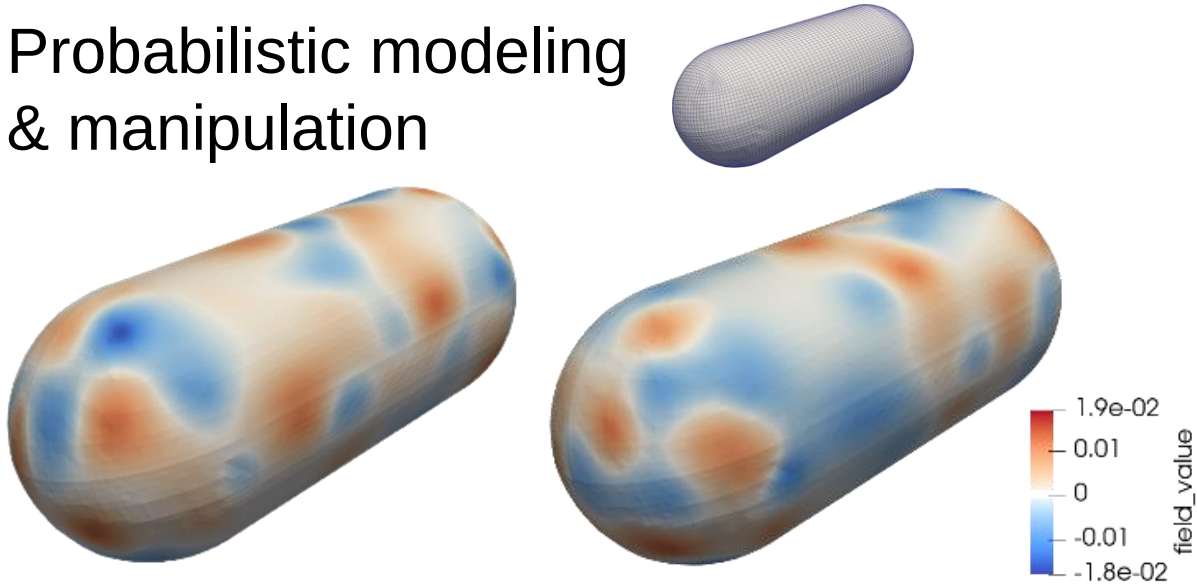
- Conversion plugins



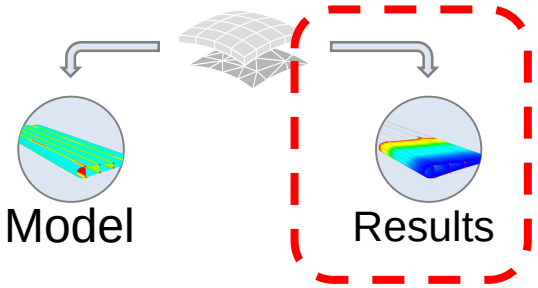
- Idealization & theory-independent model generation



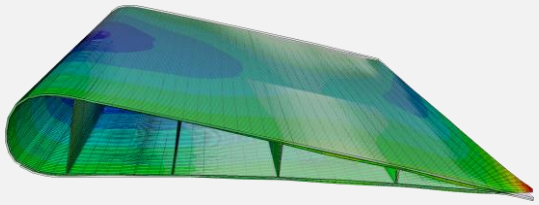
- Probabilistic modeling & manipulation



Implementation
Result I/O



Input data

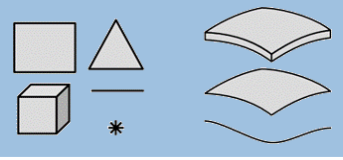


In memory

Preprocessing → *vampire* → Postprocessing

FE-Resultdata handling

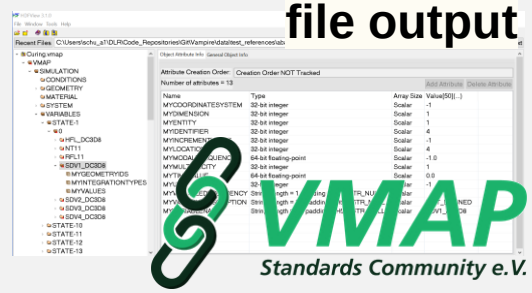
Mesh discretization storage

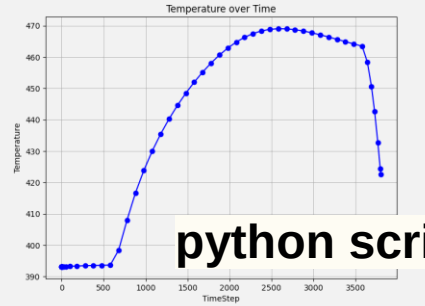


file data access

API	*simples*	*medcoupling*	*pynastran,h5py*	*?*

file output



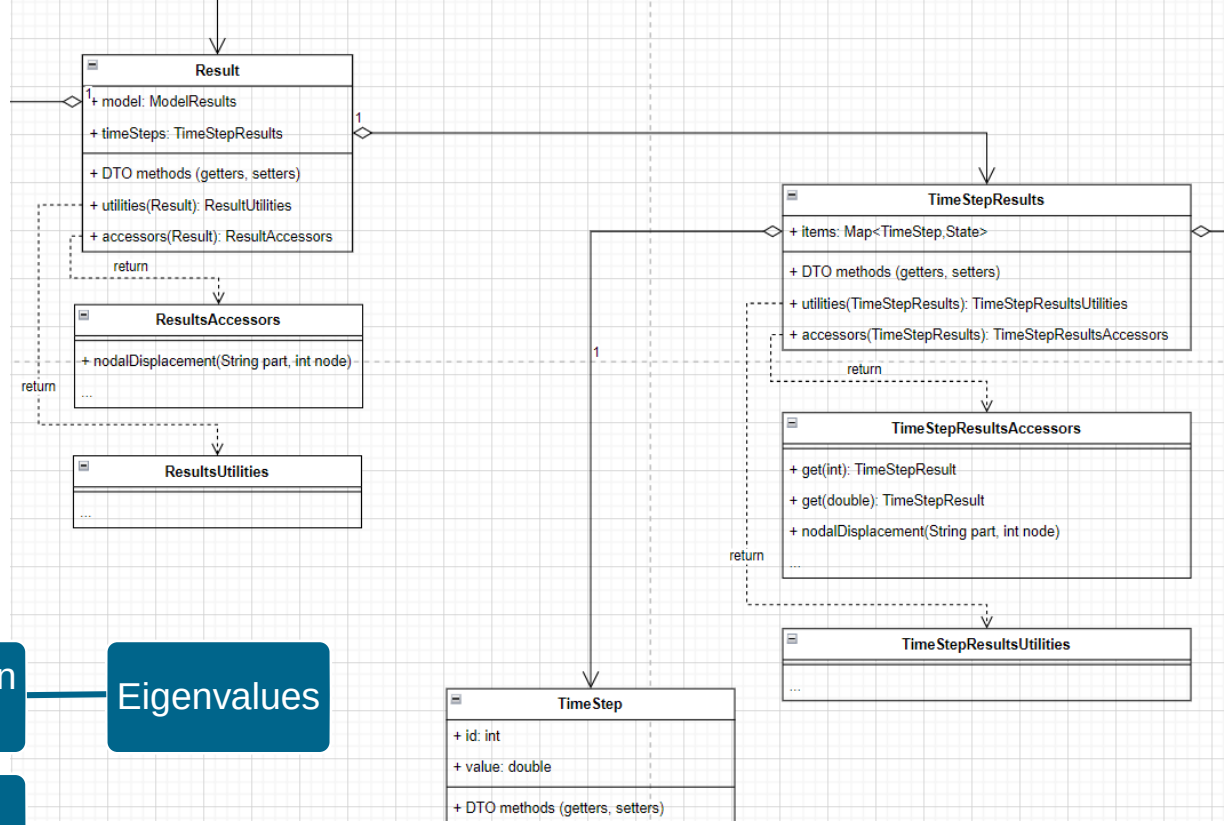


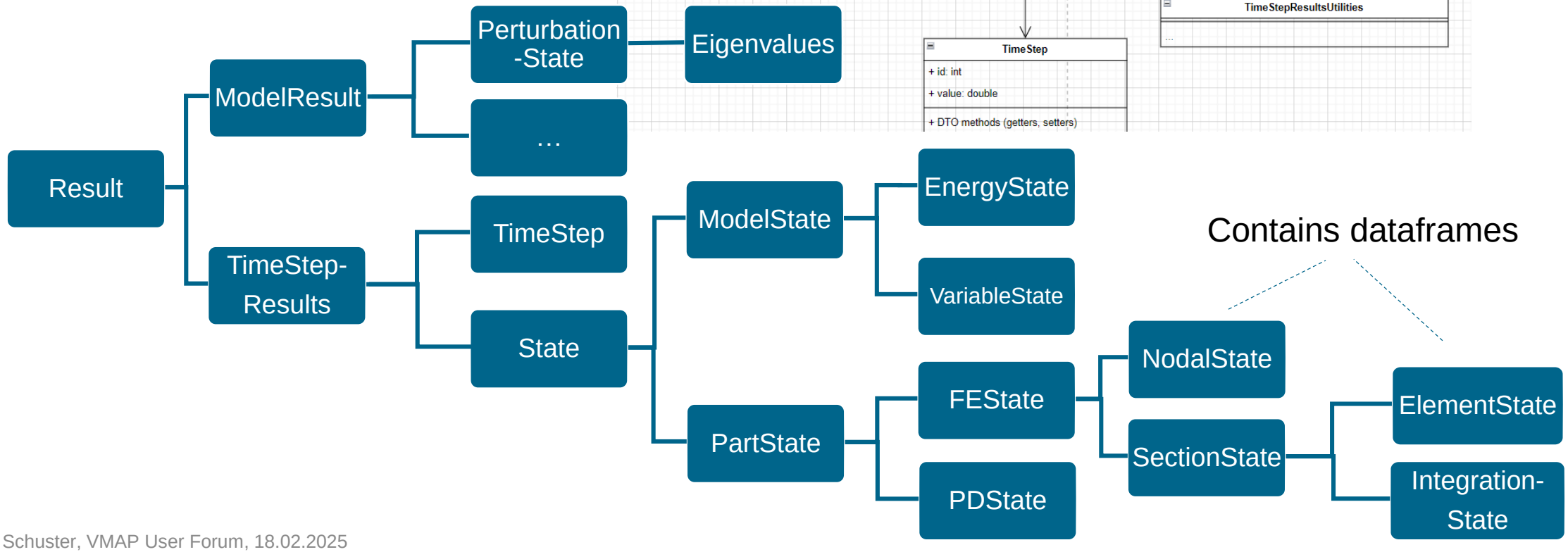
python scripting

SIMULIA ABAQUS · b2000 pro++ · code_aster · MSC Nastran™ · • • •

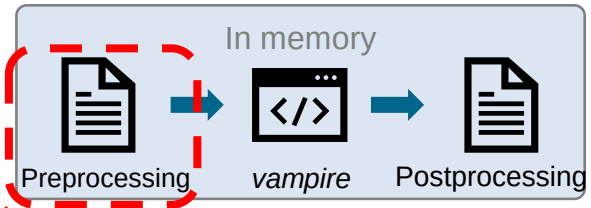
Implementation
Results – Data structure

- Simplified from UML
- Based on user stories
- Expandable





Implementation
Results – Access Abaqus data



- Investigated open source libraries:
  - PyAbaqus
  - AbaPy
  - pylife-odbclient
  - In-house tools
- Implementation features
  - Data acquisition by means of Abaqus python API
  - Read all available result types (curr. elemental, nodal) @ timesteps
  - Subselection possible
- Data storage: simple datatypes (list,dict)
- Serialization → pickle protocol in python


```
{('DC3D8', 'HFL'): {'bulk': {},
                    'data': array([[ 1.00000000e+00,  1.00000000e+00,  1.00000000e+00,
                                    -2.75148878e-14,  7.42573562e-16,  1.64136949e-07],
                                   [ 1.00000000e+00,  2.00000000e+00,  1.00000000e+00,
                                    -5.13516164e-14,  4.74825664e-15,  1.64136949e-07],
                                   [ 1.00000000e+00,  3.00000000e+00,  1.00000000e+00,
                                     1.54062164e-14,  7.42573562e-16,  1.64136949e-07],
                                   [ 1.00000000e+00,  4.00000000e+00,  1.00000000e+00,
                                    -1.16332725e-13,  4.74825664e-15,  1.64136949e-07],
                                   [ 1.00000000e+00,  5.00000000e+00,  1.00000000e+00,
                                    -2.75148878e-14,  2.55198914e-15,  1.64136949e-07],
                                   [ 1.00000000e+00,  6.00000000e+00,  1.00000000e+00,
                                    -5.13516164e-14,  2.82340497e-15,  1.64136949e-07],
                                   [ 1.00000000e+00,  7.00000000e+00,  1.00000000e+00,
                                    -3.91544097e-14,  2.55198914e-15,  1.64136949e-07],
                                   [ 1.00000000e+00,  8.00000000e+00,  1.00000000e+00,
                                    -1.16332725e-13,  2.06206529e-15,  1.64136949e-07]]),
                    'header': ['ELEM_ID', 'LOCAL_INTPTNUM', 'SECTNUM', 'HFL1', 'HFL2', 'HFL3'],
                    'meta': {'location': 'INTEGRATION_POINT'},
                    'type': ''}}
```

element type and elemental result name

Raw data

column description

Code Example

```python
from vampire.datamodel.simulation import Simulation
import seaborn as sns
inputFile = os.path.join(
    os.path.dirname(__file__),"Test_CureModel_HT.odb"
)
sim = Simulation()
sim.readModel(inputFile)
sim.readResults(inputFile)

data = sim.results.getTimeSeriesNodalData(
    loadCaseName="Curing",
    partName="PART-1-1",
    dataSetName="NT11",
    identifiers=[1.0, 2.0],
    timeStepMin=0,
    timeStepMax=-1,
)

sns.lineplot(
    data,
    x="timeStep",y="COL1",
    hue="identifier",palette="viridis"
)

sim.export(os.path.dirname(__file__),filetype="vmap")
```

- Simulation object definition
- Define input files with mesh and result data
- Required result parser inferred from file extension (explicit solver name possible)

Code Example

```python
from vampire.datamodel.simulation import Simulation
import seaborn as sns
inputFile = os.path.join(
    os.path.dirname(__file__),"Test_CureModel_HT.odb"
)
sim = Simulation()
sim.readModel(inputFile)
sim.readResults(inputFile)

data = sim.results.getTimeSeriesNodalData(
    loadCaseName="Curing",
    partName="PART-1-1",
    dataSetName="NT11",
    identifiers=[1.0, 2.0],
    timeStepMin=0,
    timeStepMax=-1,
)

sns.lineplot(
    data,
    x="timeStep",y="COL1",
    hue="identifier",palette="viridis"
)

sim.export(os.path.dirname(__file__),filetype="vmap")
```

- Use of generic access routines
- Selected results as ordinary *pandas* DataFrames

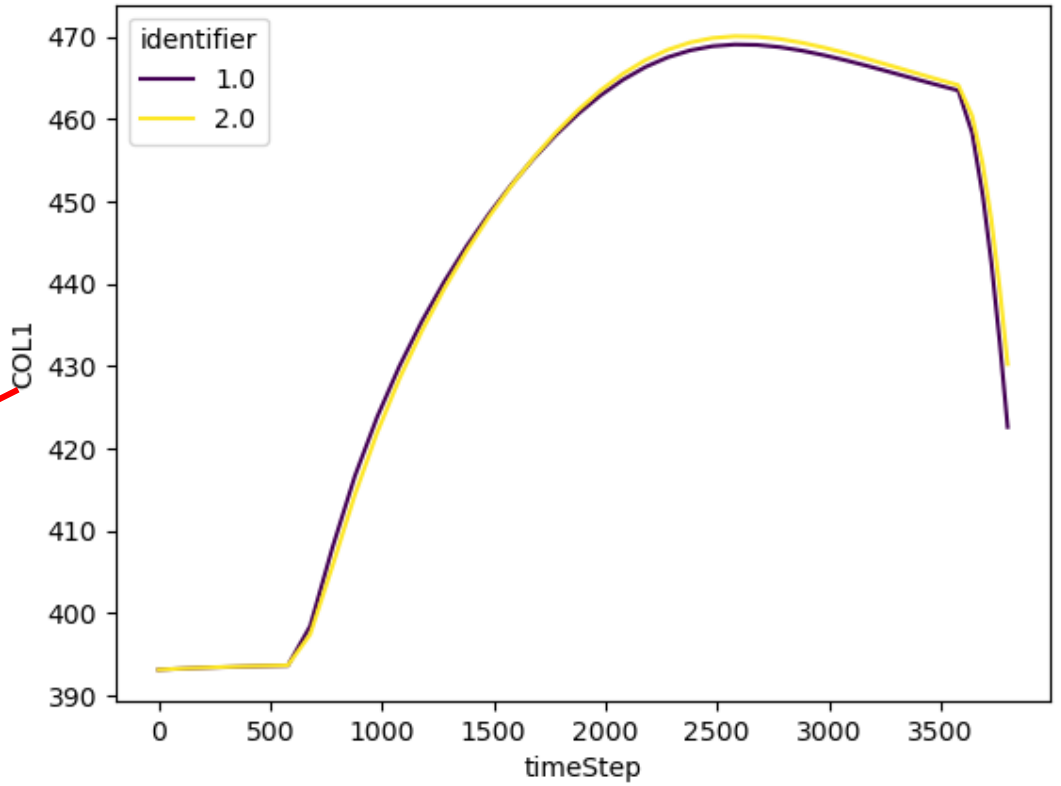
	identifier	timeStep	COL1
0	1.0	0.000000	393.149994
1	2.0	0.000000	393.149994
2	1.0	0.100000	393.150146
3	2.0	0.100000	393.150146
4	1.0	0.200000	393.150299
..	...	...	...
103	2.0	3765.070312	439.321869
104	1.0	3793.669922	424.503540
105	2.0	3793.669922	431.945251
106	1.0	3800.000000	422.618774
107	2.0	3800.000000	430.264496

[108 rows x 3 columns]

Code Example

```python
from vampire.datamodel.simulation import Simulation
import seaborn as sns
inputFile = os.path.join(
    os.path.dirname(__file__),"Test_CureModel_HT.odb"
)
sim = Simulation()
sim.readModel(inputFile)
sim.readResults(inputFile)

data = sim.results.getTimeSeriesNodalData(
    loadCaseName="Curing",
    partName="PART-1-1",
    dataSetName="NT11",
    identifiers=[1.0, 2.0],
    timeStepMin=0,
    timeStepMax=-1,
)

sns.lineplot(
    data,
    x="timeStep",y="COL1",
    hue="identifier",palette="viridis"
)

sim.export(os.path.dirname(__file__),filetype="vmap")
```

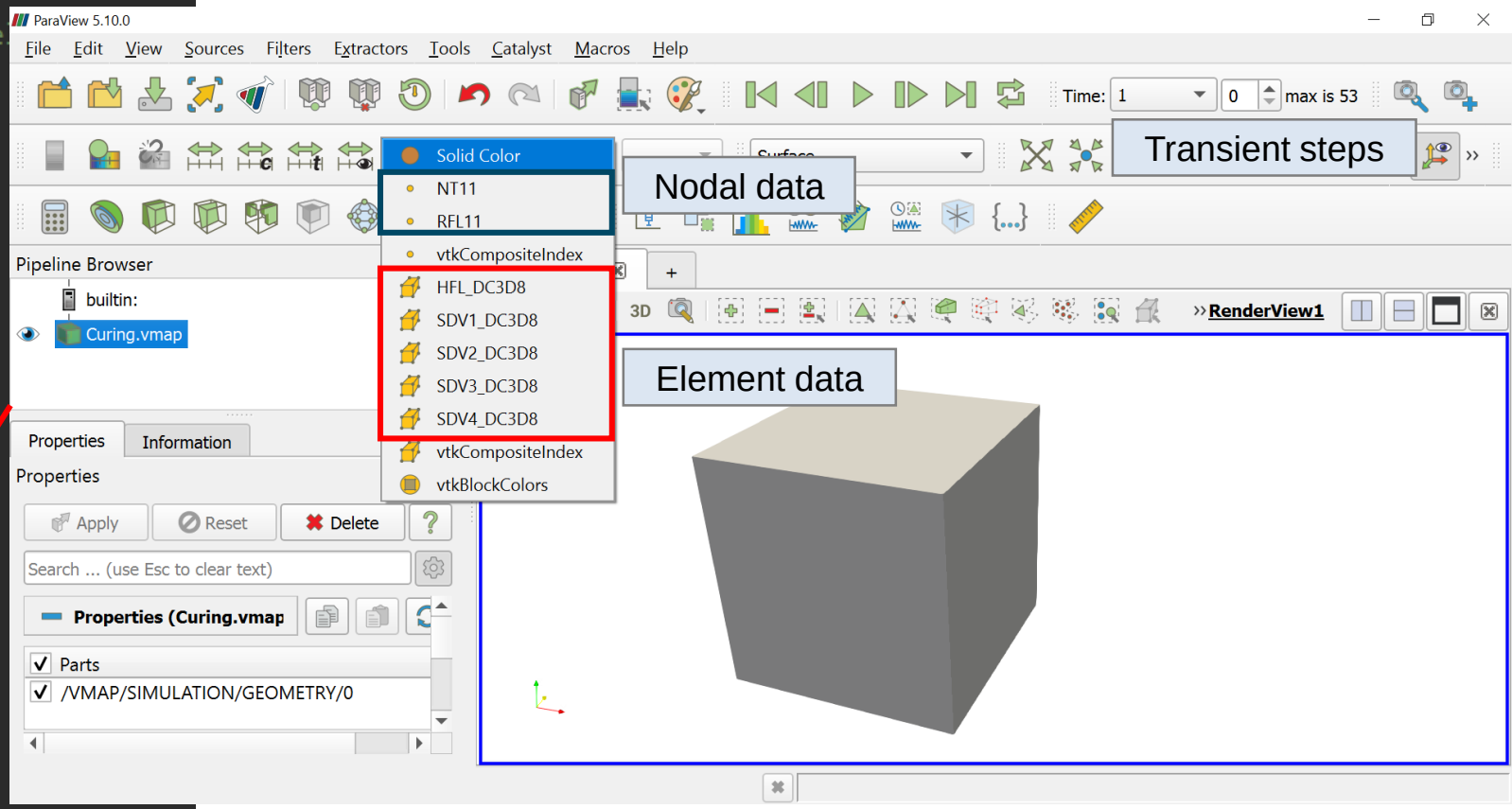
Easy use of common functions to plot data



Code Example

```python
from vampire.datamodel.simulation import Simulation
import seaborn as sns
inputFile = os.path.join(
    os.path.dirname(__file__),"Test_CureMode
)
sim = Simulation()
sim.readModel(inputFile)
sim.readResults(inputFile)

data = sim.results.getTimeSeriesNodalData(
    loadCaseName="Curing",
    partName="PART-1-1",
    dataSetName="NT11",
    identifiers=[1.0, 2.0],
    timeStepMin=0,
    timeStepMax=-1,
)

sns.lineplot(
    data,
    x="timeStep",y="COL1",
    hue="identifier",palette="viridis"
)

sim.export(os.path.dirname(__file__),filetype="vmap")
```

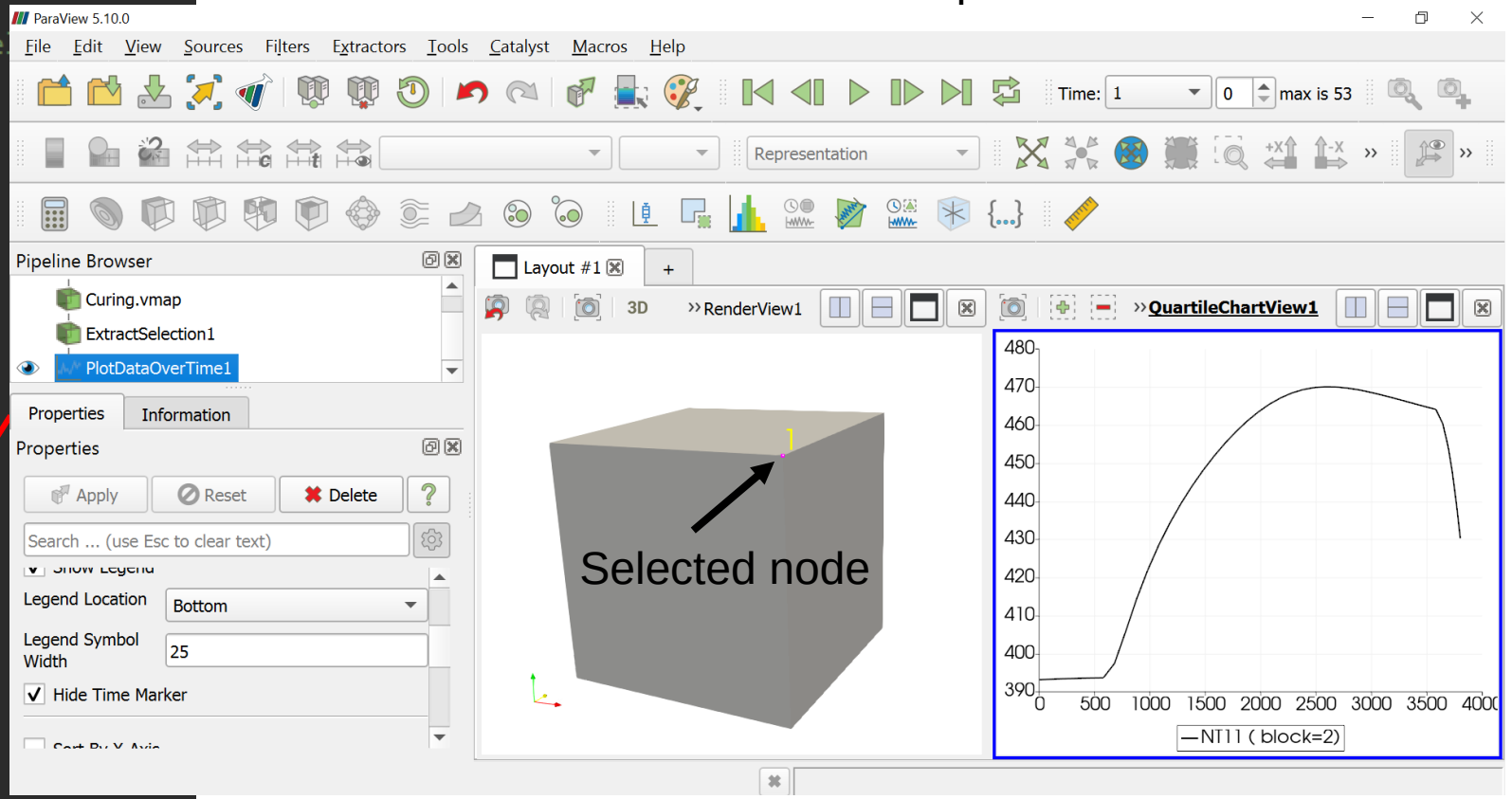
VMAP File Export



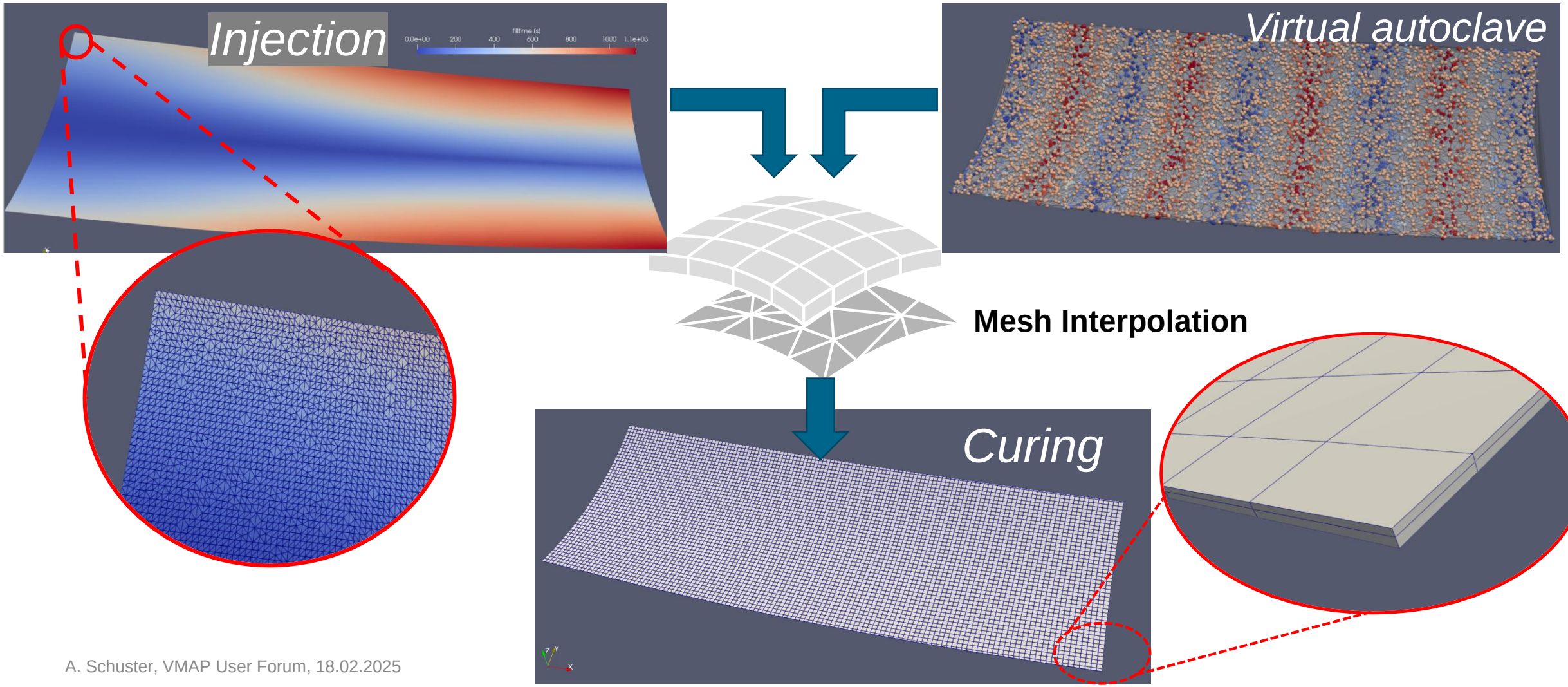
Code Example

```python
from vampire.datamodel.simulation import Simulation
import seaborn as sns
inputFile = os.path.join(
    os.path.dirname(__file__),"Test_CureMode
)
sim = Simulation()
sim.readModel(inputFile)
sim.readResults(inputFile)

data = sim.results.getTimeSeriesNodalData(
    loadCaseName="Curing",
    partName="PART-1-1",
    dataSetName="NT11",
    identifiers=[1.0, 2.0],
    timeStepMin=0,
    timeStepMax=-1,
)

sns.lineplot(
    data,
    x="timeStep",y="COL1",
    hue="identifier",palette="viridis"
)

sim.export(os.path.dirname(__file__),filetype="vmap")
```

VMAP File Export



Manufacturing Simulation
Results



Summary

- Tools and processes for simulation-based assessment of composite structure manufacturing processes developed at DLR
- Extensible storage approach for discretization based FE model and result data
- Pre- and post-processing interfaces to various tools and solvers available
- seamless integration of VMAP standard into tool-chain of selected project usecase
- Next steps: interpolation of data from injection and autoclave simulation to target CFRP structure required for curing simulation

THANK YOU FOR YOUR ATTENTION

Contact

Andreas Schuster

German Aerospace Center
Institute of Lightweight Systems
Department of Structural Mechanics

Lilienthalplatz 7
38108 Braunschweig

☎ +49 (0)531 295-2778
🖷 +49 (0)531 295-2232
✉ andreas.schuster@dlr.de
💻 www.dlr.de/sy

Gefördert durch:

Bundesministerium für Wirtschaft und Klimaschutz

aufgrund eines Beschlusses des Deutschen Bundestages

PredictECO