KU LEUVEN

TECHPED

Technically feasible positive energy districts

IBPSA Task 3 Meeting
Alternative wrapped FMU generation

Contributing towards identifying feasible, effective and economical solutions, focusing both on system lay-out and operation/control of Positive Energy Districts (PEDs)

Cas Bex

August 5th, 2024

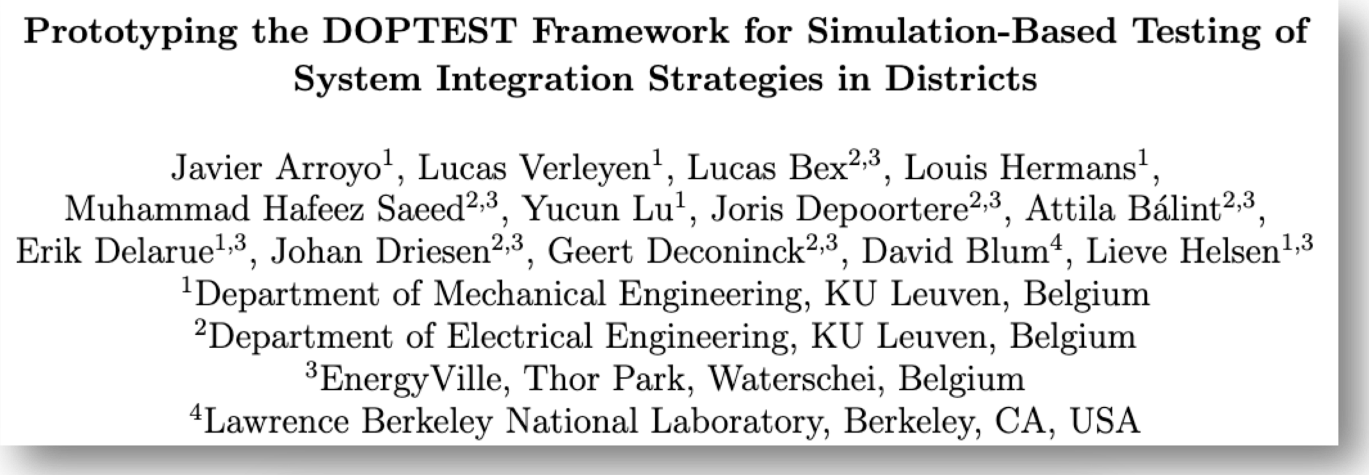
Content

- Background
- Alternative build strategy for wrapped FMUs

Background

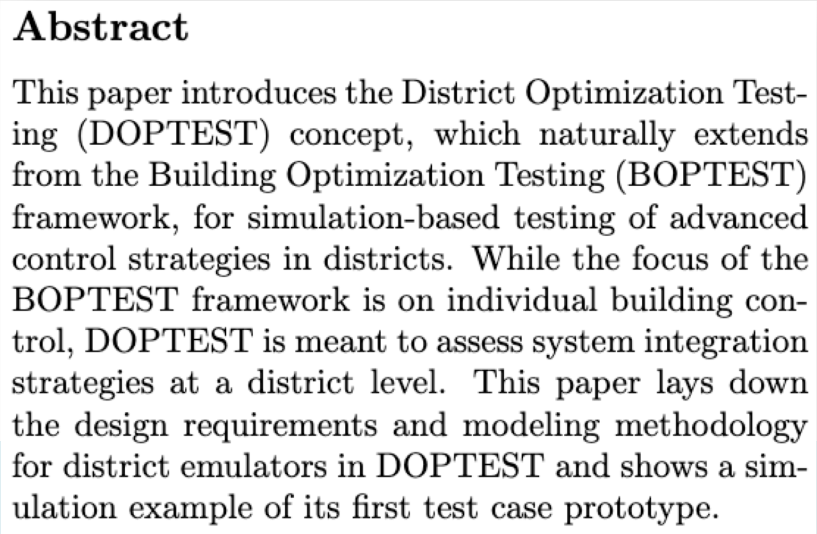
DOPTEST

BS 2023
4 - 6 SEPT
SHANGHAI



Prototyping the DOPTEST Framework for Simulation-Based Testing of System Integration Strategies in Districts

Javier Arroyo[1], Lucas Verleyen[1], Lucas Bex[2,3], Louis Hermans[1],
Muhammad Hafeez Saeed[2,3], Yucun Lu[1], Joris Depoortere[2,3], Attila Bálint[2,3],
Erik Delarue[1,3], Johan Driesen[2,3], Geert Deconinck[2,3], David Blum[4], Lieve Helsen[1,3]
[1]Department of Mechanical Engineering, KU Leuven, Belgium
[2]Department of Electrical Engineering, KU Leuven, Belgium
[3]EnergyVille, Thor Park, Waterschei, Belgium
[4]Lawrence Berkeley National Laboratory, Berkeley, CA, USA

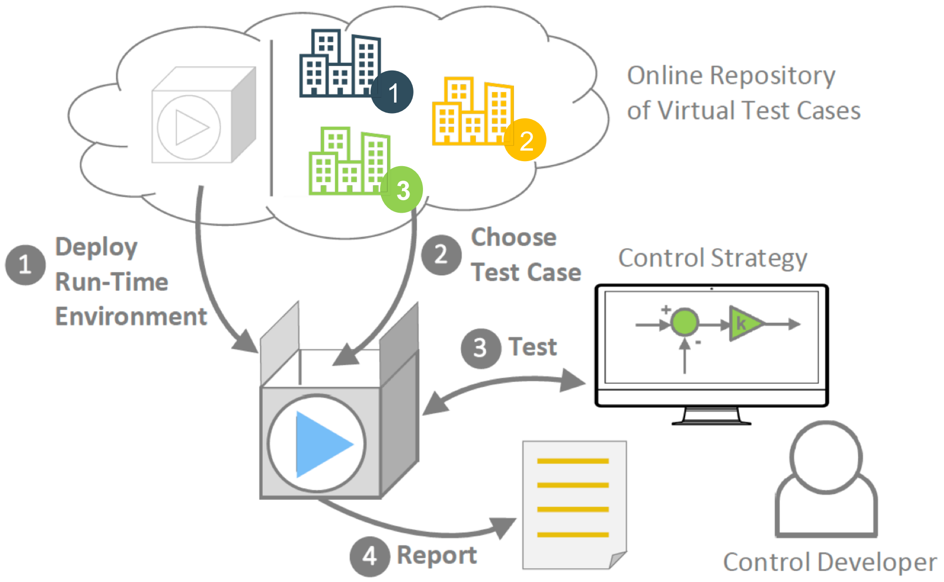


Abstract

This paper introduces the District Optimization Testing (DOPTEST) concept, which naturally extends from the Building Optimization Testing (BOPTEST) framework, for simulation-based testing of advanced control strategies in districts. While the focus of the BOPTEST framework is on individual building control, DOPTEST is meant to assess system integration strategies at a district level. This paper lays down the design requirements and modeling methodology for district emulators in DOPTEST and shows a simulation example of its first test case prototype.

DOPTEST

*From **B**OPTEST to → **D**OPTEST = District optimization testing framework*



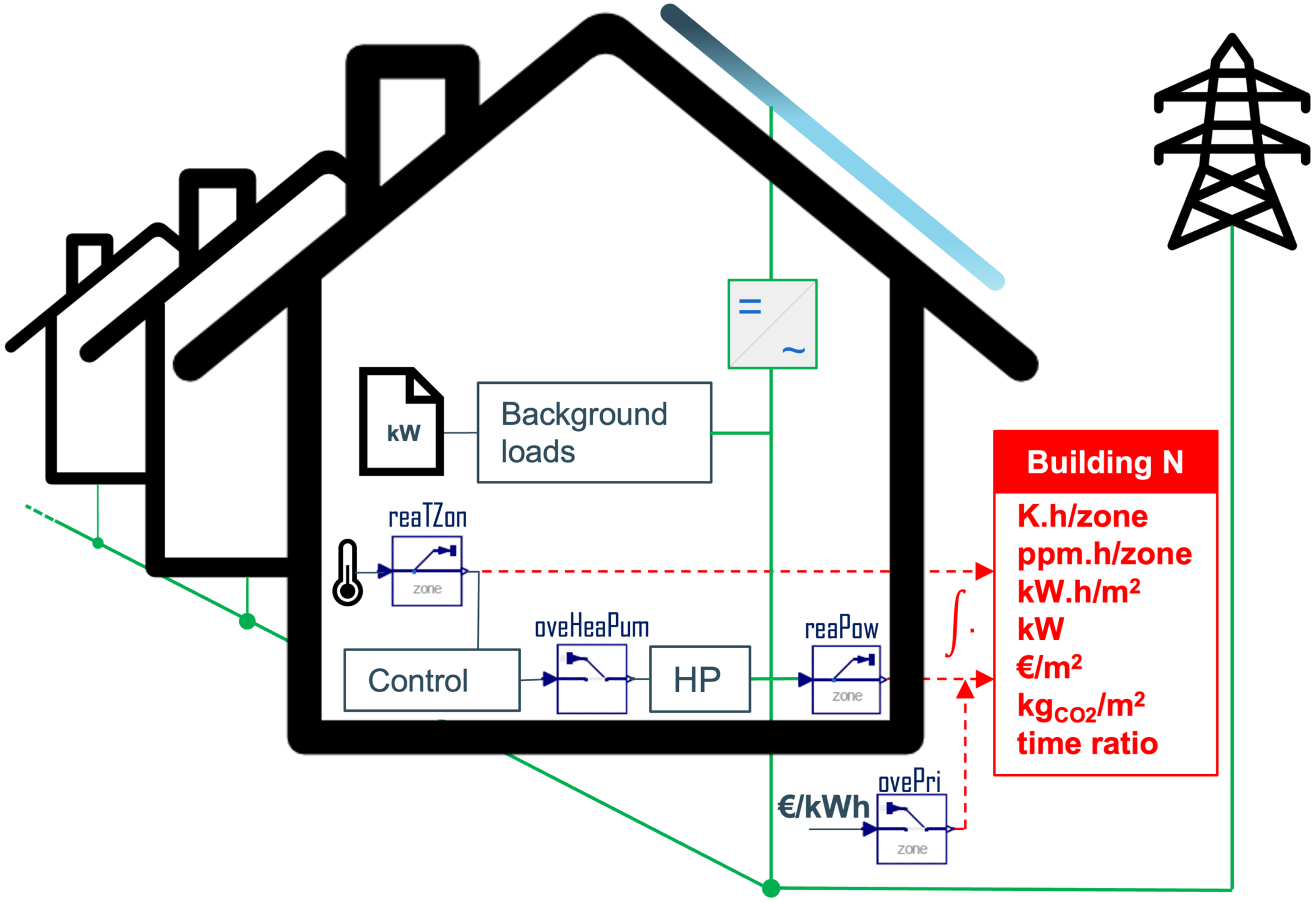
Main stakeholders :

- Industry developers
- Algorithm researchers
- **Aggregators**
- **Grid operators**

https://github.com/ibpsa/project1-boptest

Presented at the BS2023

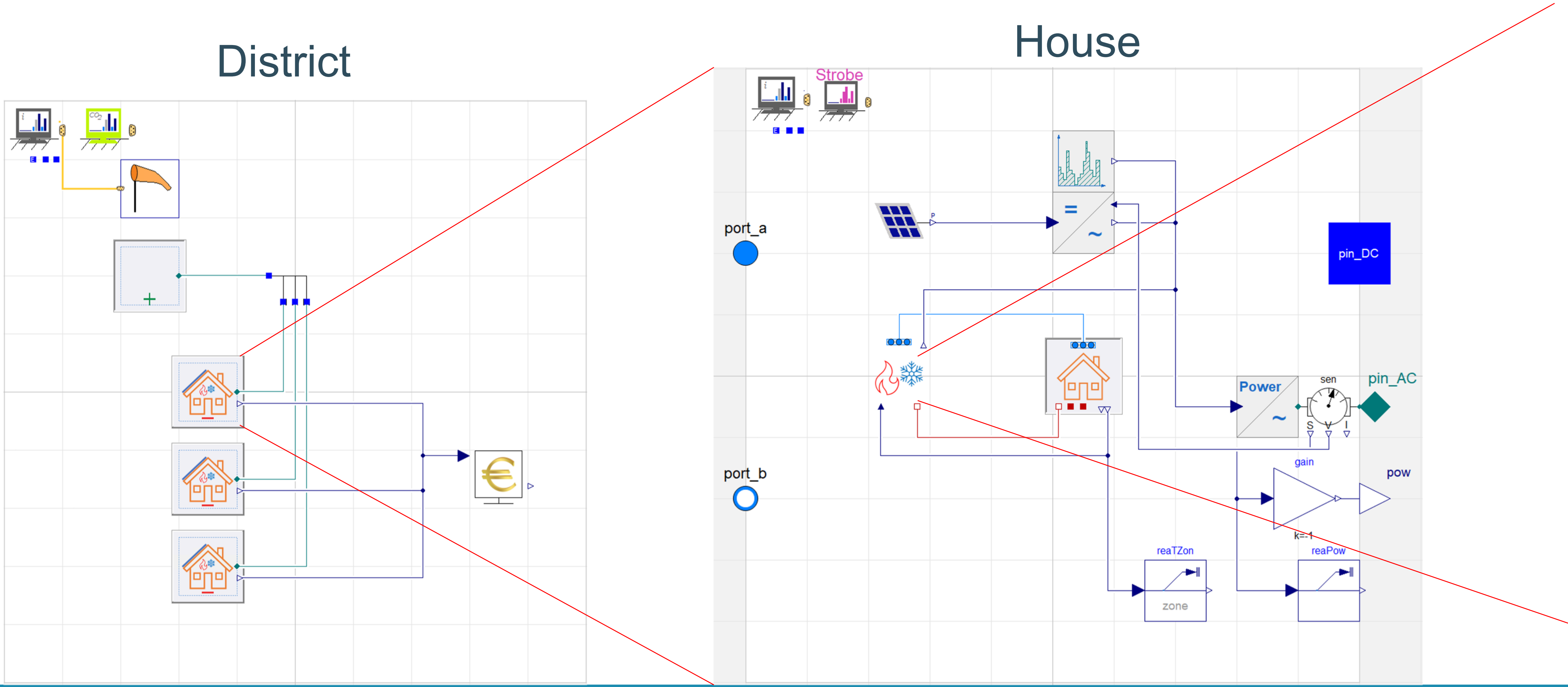
Tiny Cluster



Models available at:

https://gitlab.kuleuven.be/positive-energy-districts/moped

Tiny Cluster

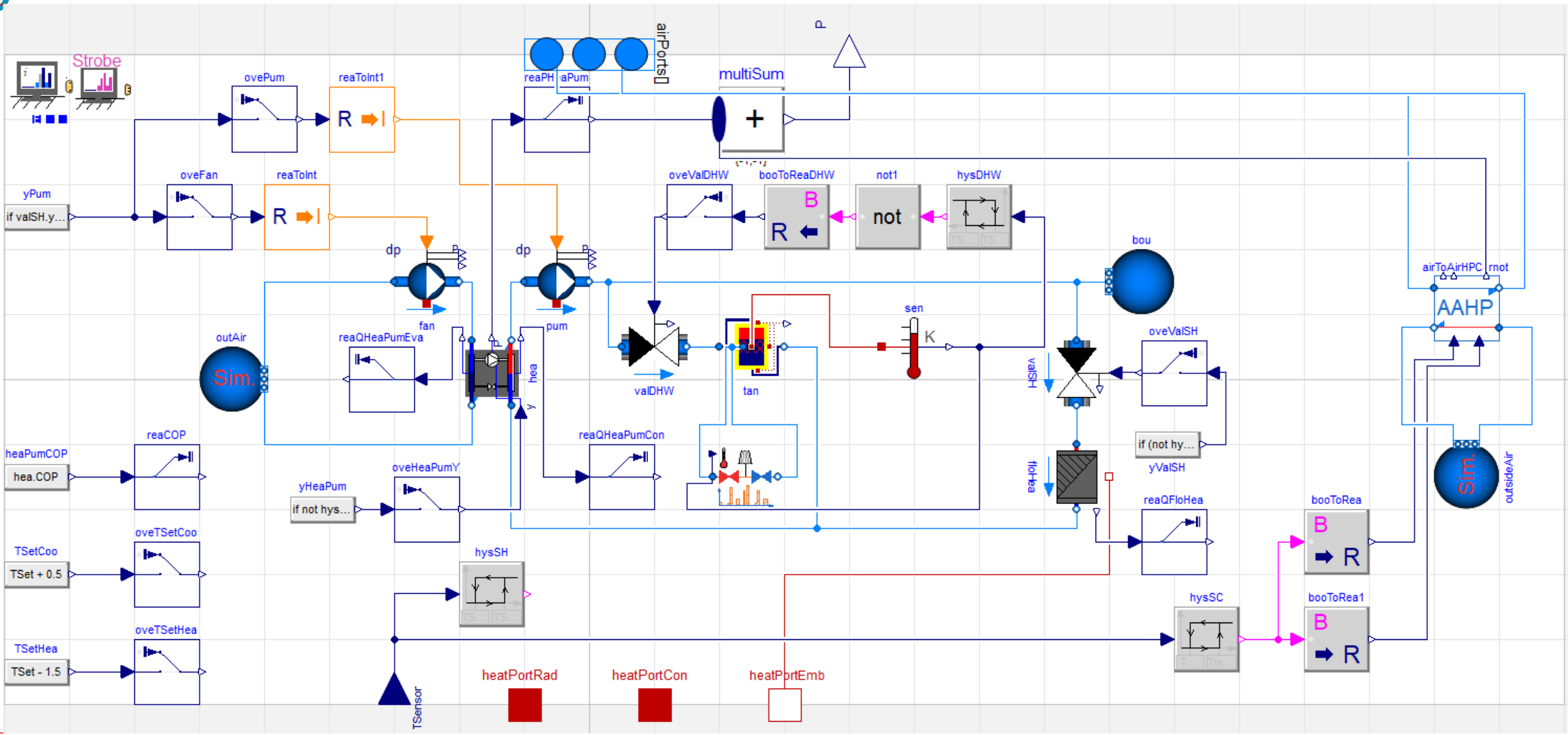


District

House

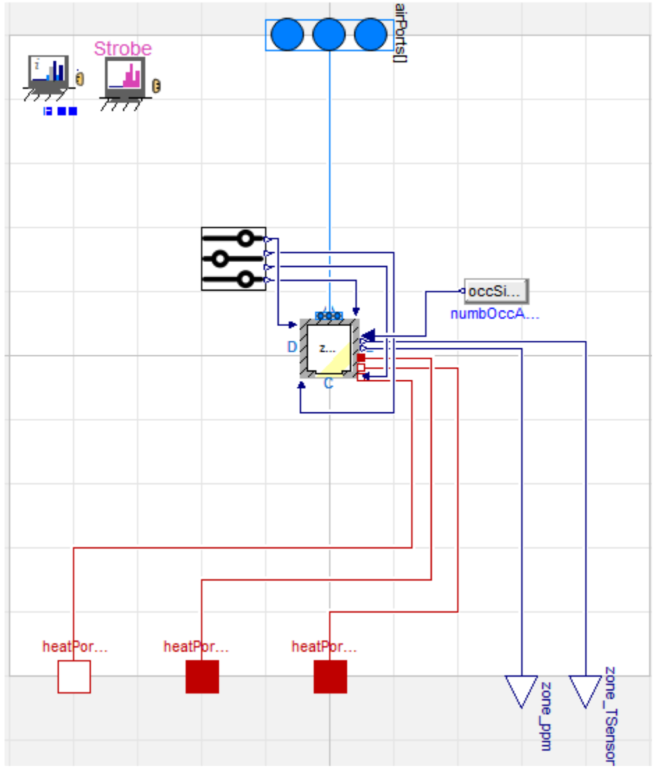
Tiny Cluster

HVAC



TECHPED
Technically feasible positive energy districts
Energy Ville
KU LEUVEN

Tiny Cluster



IDEAS Zone template

Alternative Build Strategy for Wrapped FMUs

Why an Alternative Build Strategy?

- OpenModelica is **slow** to compile FMUs
  - Faster builds are possible

- OpenModelica FMUs currently have (critical) **memory leaks**
  - Full-year simulations are prohibitive
  - Shorter typical two-week DOPTEST simulations are ok

How To Build a Test Case

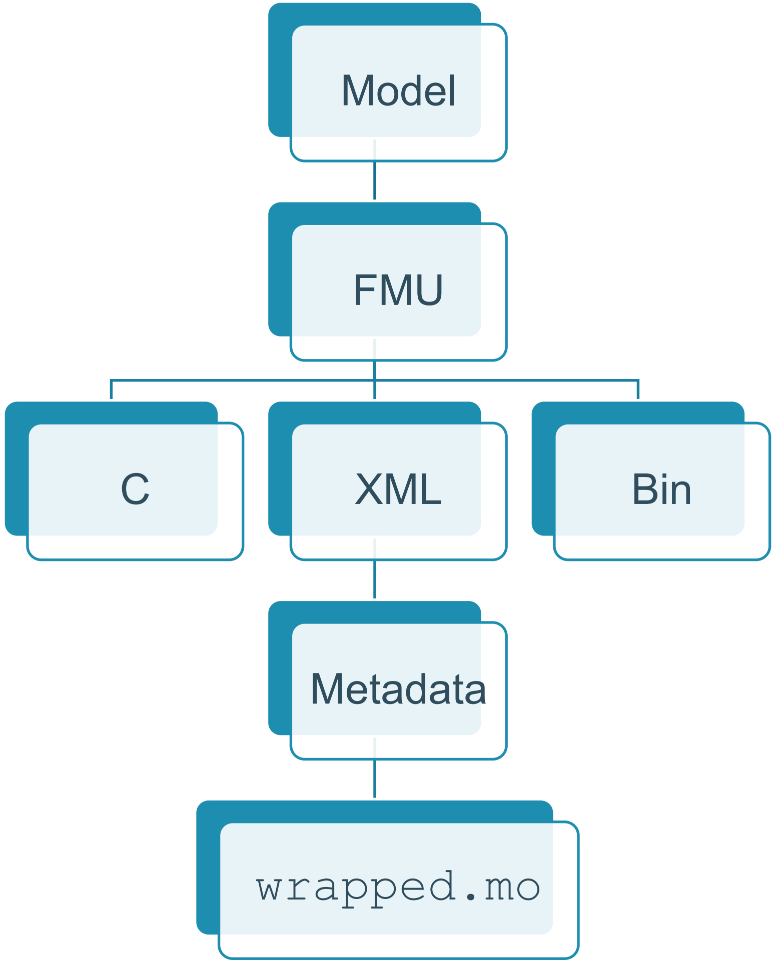
- Wrap original model using boptestRead/boptestWrite blocks
- Tag KPI variables
- Create test case configuration
- Obtain boundary condition data

How To Build a Test Case

- **Wrap original model using boptestRead/boptestWrite blocks**
- Tag KPI variables
- Create test case configuration
- Obtain boundary condition data

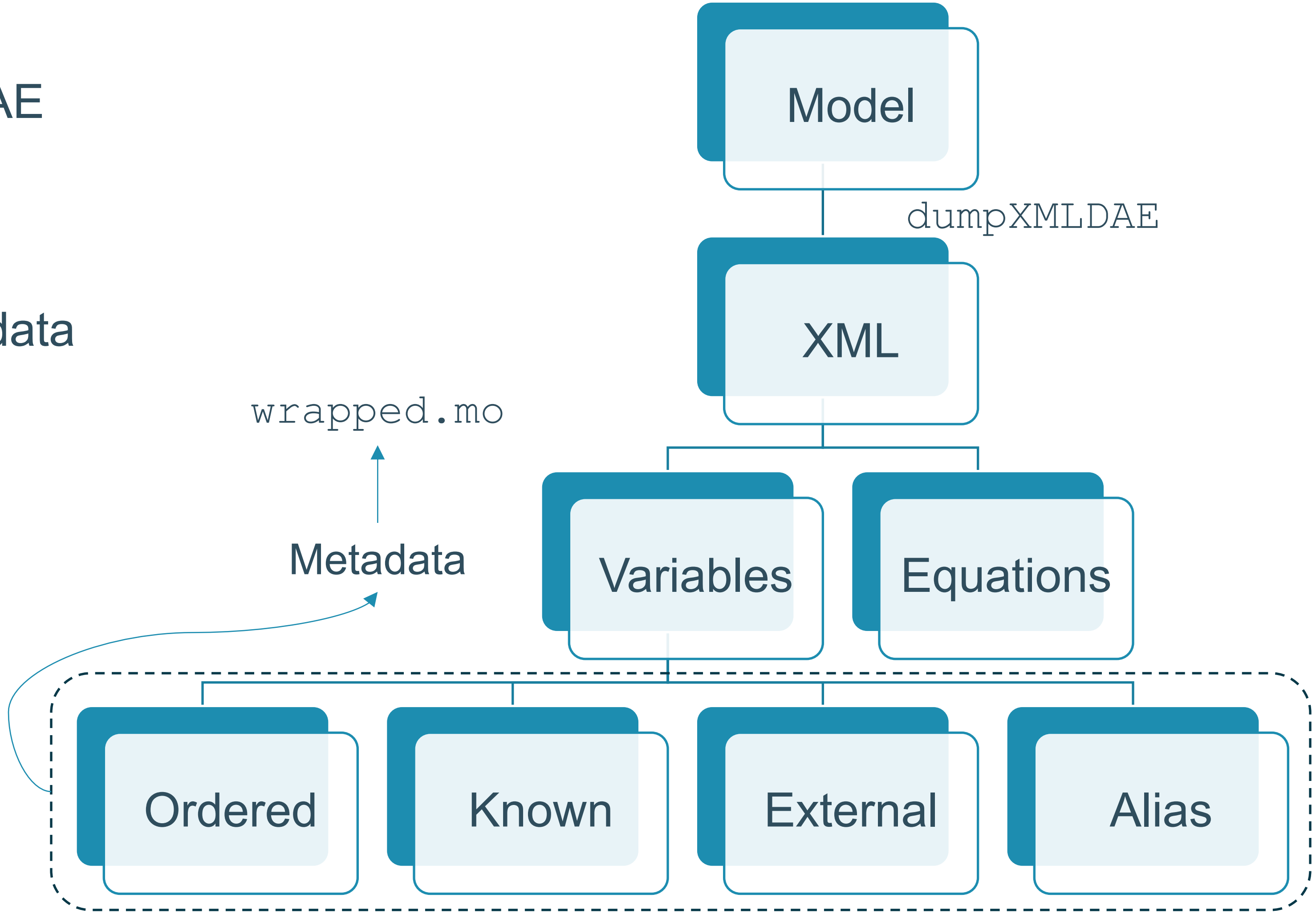
BOPTEST parsing creates intermediate FMU

1. Export original model into an FMU
2. Search for all boptestOverwrite and boptestRead blocks
3. Record metadata
4. Create wrapped model



Our parser works with XML descriptors

1. Obtain XML description of model DAE
2. Iterate over variables in XML
    - If read/write block: record metadata
3. Create wrapped model

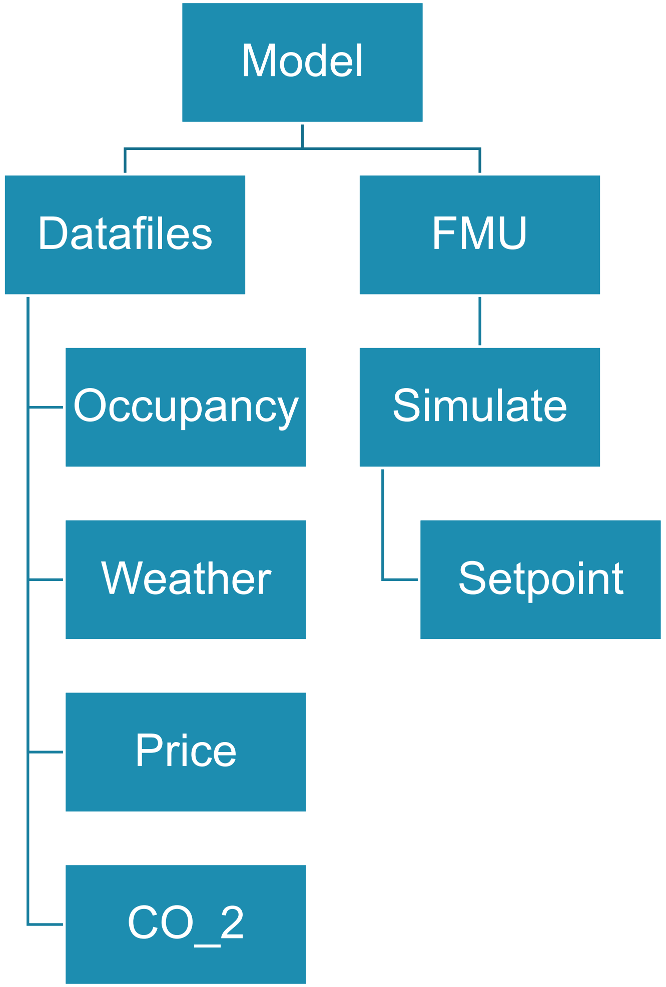


How To Build a Test Case

- Wrap original model using boptestRead/boptestWrite blocks
- Tag KPI variables
- Create test case configuration
- **Obtain boundary condition data**

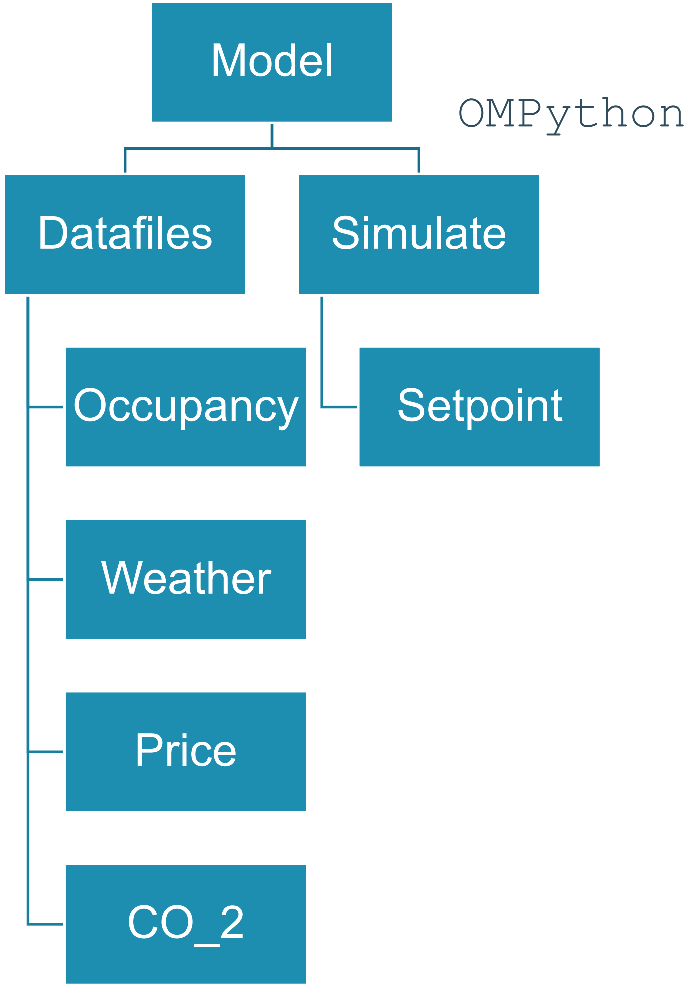
Boundary conditions through simulation: BOPTEST

- BOPTEST simulates original FMU to obtain temperature setpoints



Simulating original model gives same results

- OMPython allows to easily get simulation results directly from OM



Conclusion

- Use `dumpXMLDAE` to avoid original FMU compilation → faster
- Use `OMPython` to avoid simulating FMUs long-term → memory leak avoided

Questions?

```
$ make wrapped -n -B
# create build directory
mkdir -p build/Full
# fill in model name in template scripts
sed -b 's~MODELPATH~../../src/Districts/Full.mo~' tools/fmu_template.mos | \
 sed -b "s~MODEL~Full~" > build/Full/compile_original_fmu.mos
sed -b 's~MODELPATH~../../src/Districts/Full.mo~' tools/xml_template.mos | \
 sed -b "s~MODEL~Full~" > build/Full/dump_xml.mos
# get the DAE XML description
cd build/Full && omc dump_xml.mos | tee tmpdump.txt
# catch the XML filename from omc output and change it
mv $(tail -n 1 build/Full/tmpdump.txt | grep -oe '".*"' | tr -d '"') build/Full/dump.xml
rm build/Full/tmpdump.txt
# parse XML to get metadata
python tools/dae_xml_parser.py build/Full/dump.xml -o build/Full/wrapped.mo -m Full
# fill in model name in template scripts
sed -b 's~loadFile(\"\(.*/\)\(\w\+\.mo\)\")~loadFile(\"\1\2\");\nloadFile(\"wrapped.mo\")~’ \
 build/Full/compile_original_fmu.mos | sed -b "s~FMU(\([a-zA-Z0-9\.]\+\)~FMU(wrapped~" > \
 build/Full/compile_wrapped_fmu.mos
# get precalculated boundary condition/config files from online repo
curl “myurl/resources.zip" -o build/Full/resources.zip
unzip build/Full/resources.zip -d build/Full/resources
# compile wrapped FMU and zip the boundary condition data with it
cd build/Full && omc compile_wrapped_fmu.mos -d=evaluateAllParameters && \
  (zip -d wrapped.fmu \*.csv || true) && zip wrapped.fmu -u  resources/*
```

Thanks!

C2 project

TECHPED

TECHPED
Technically feasible positive energy districts

Energy Ville

KU LEUVEN