TEASER+ : CityGML EnergyADE v1.0 implementation

3rd Expert Meeting IBPSA Project 1, Aachen

Avichal Malhotra, M.Sc

Maksim Shamovich, B.Sc

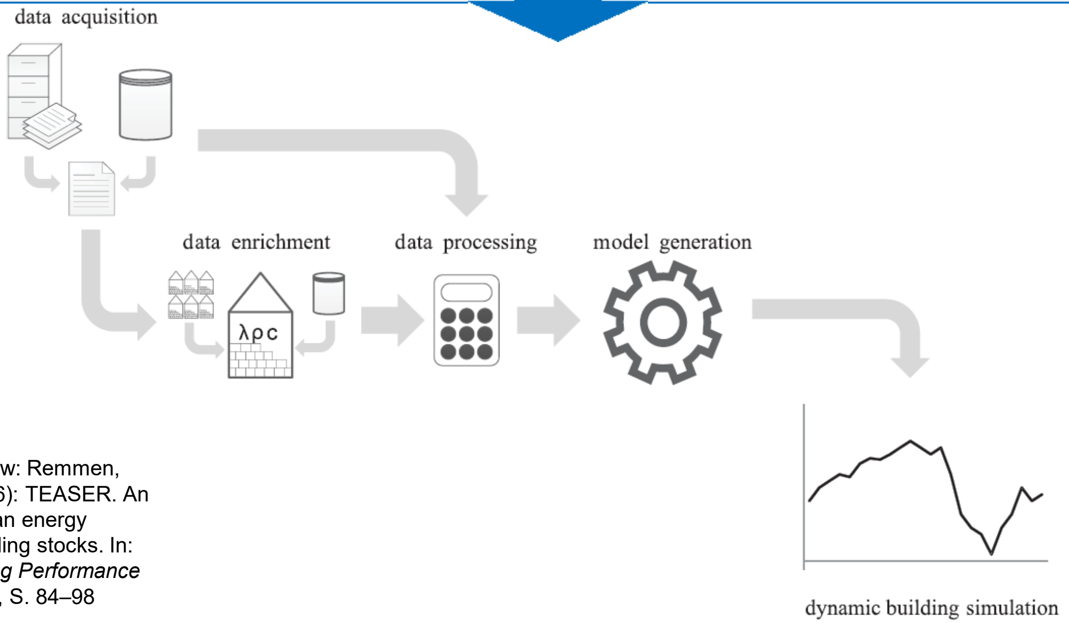
Institute of Energy Efficiency and Sustainable Building, E3D

RWTH Aachen University

04.04.2019

TEASER / AixLib: Basics

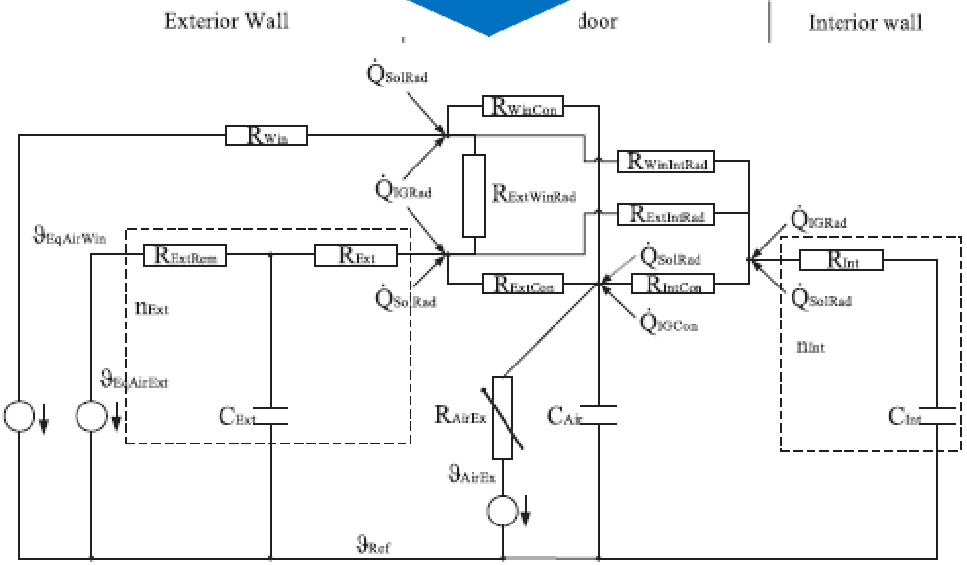
- Open-Source **Urban** energy performance analysis tool implemented in python
- **API** between input processing and simulation engine
- Focus on **automation**, quick parametrizations and **high volume throughput**, with minimal input requirements
- Integrated material, building elements and use conditions libraries for the archetype enrichment process
- generates the Modelica models for various Libraries (AixLib, IDEAS…)



TEASER Workflow: Remmen, Peter. et al. (2016): TEASER. An open tool for urban energy modelling of building stocks. In: *Journal of Building Performance Simulation* 11 (1), S. 84–98

TEASER / AixLib: Basics

- Generated model intended for AixLib library in **Dymola**
- Reduced/Lower Order Model
  - Based on VDI 6007
  - Single homogeneous thermal zone for each building
- Outer walls 2R1C models (All lumped together or base / roof separate)
- Interior Walls 1R1C (only heat storage)
- Windows modeled separate



Reduced order Model - Remmen, Peter. et al. (2016): TEASER. An open tool for urban energy modelling of building stocks. In: *Journal of Building Performance Simulation* 11 (1), S. 84–98

Starting Point for TEASER+

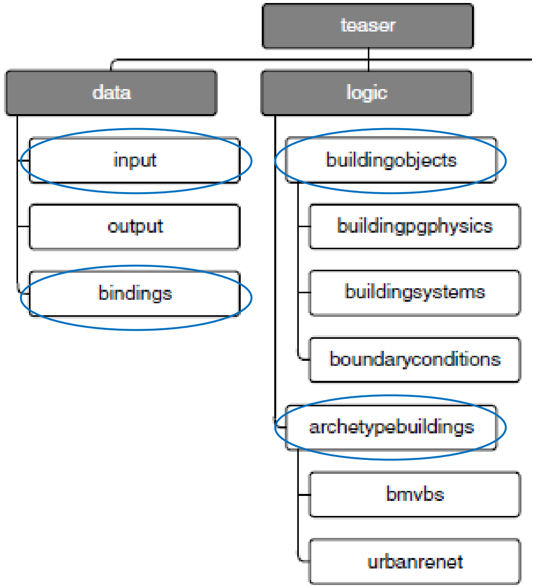
- Started with TEASER v.0.54
- Only allowed CityGML input files of LoD1 and LoD2
- PyXB bindings for EnergyADE v0.7 for output
- Integrated Databases/Templates:
  - BMVBS (non-residential)
  - IWU (residential)
  - Urbanrenet
- Model generation using AixLib v.0.52

TEASER
Tool for Energy Analysis and Simulation for Efficient Retrofit

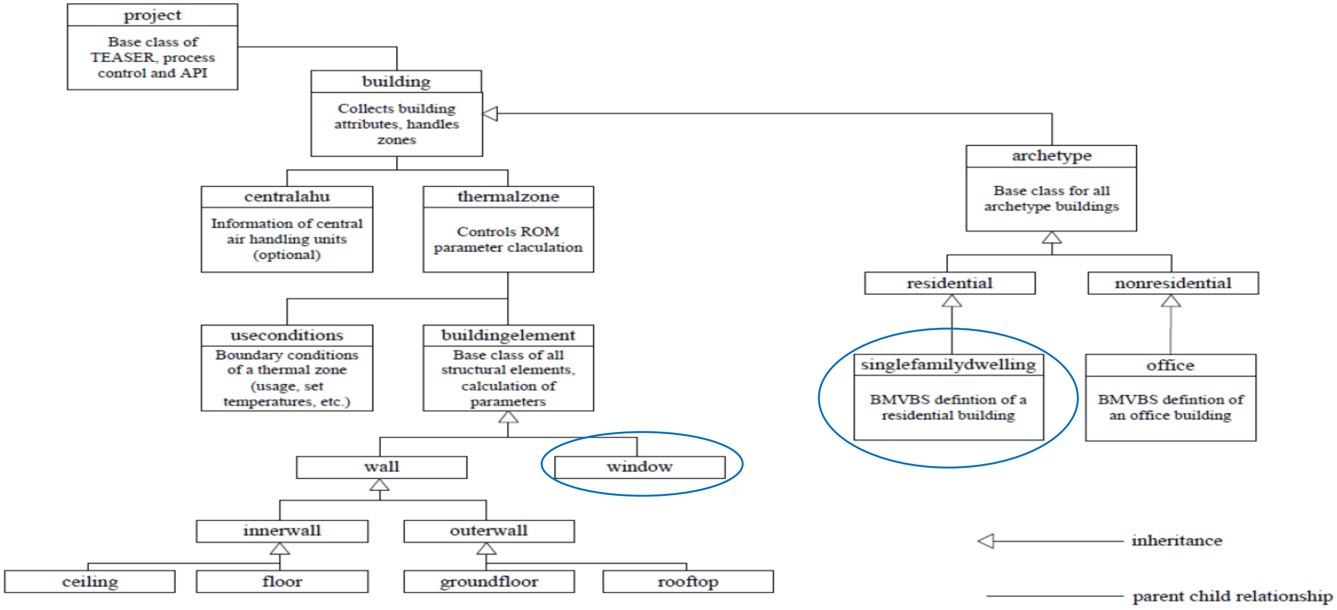
Developed by: RWTH Aachen University, E.ON Energy Research Center, Institute for Energy Efficient Buildings and Indoor Climate

TEASER+: Extensions

- Extending the import capabilities:
  - EnergyADE v. 1.0 as input for LoD 0-3 ✓
  - LoD0 (Creating LoD1 model) ✓
  - LoD3 (Windows and Doors) ✓
  - LoD4 (Inner Wall area) (✓)

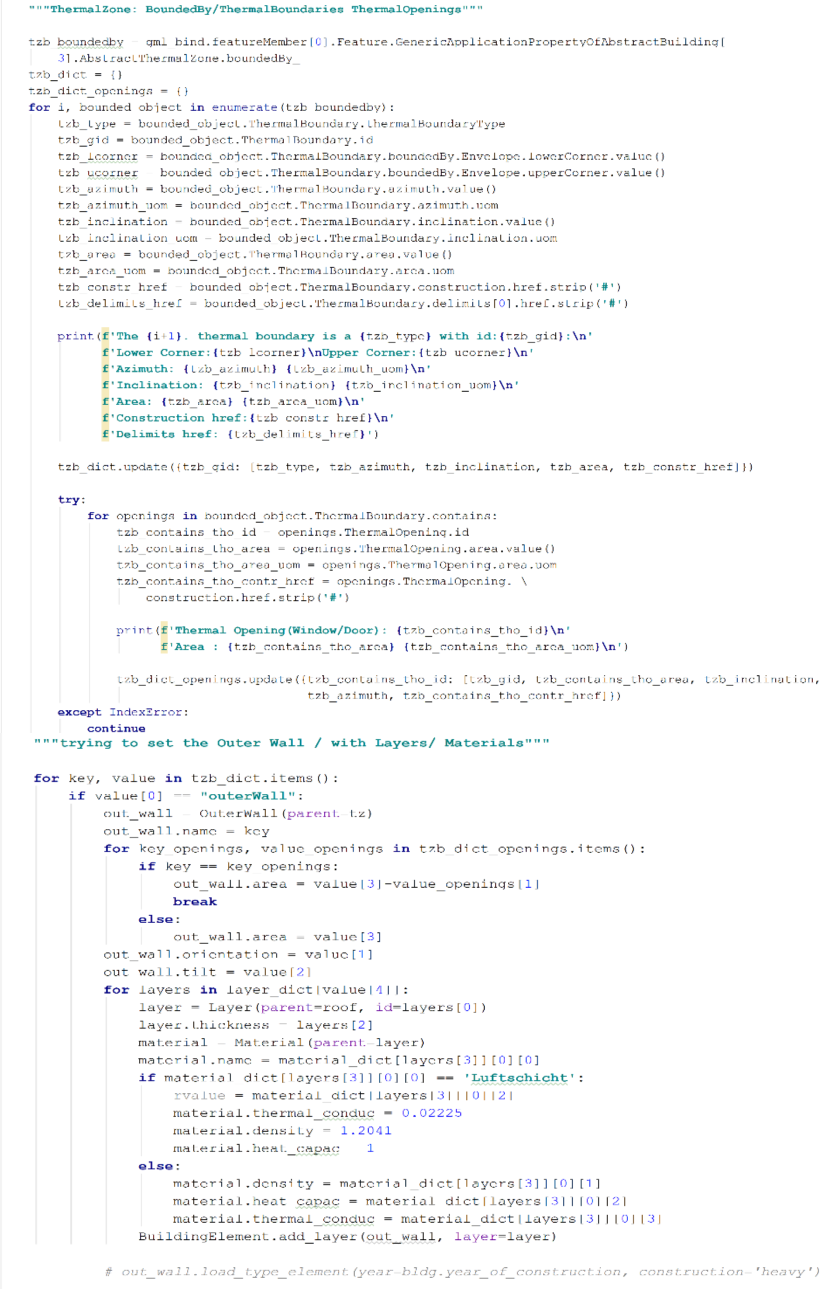


TEASER module structure - Remmen, Peter. et al. (2016): TEASER. An open tool for urban energy modelling of building stocks. In: *Journal of Building Performance Simulation* 11 (1), S. 84–98


project
Base class of TEASER, process control and API
building
Collects building attributes, handles zones
centralahu
Information of central air handling units (optional)
thermalzone
Controls ROM parameter claculation
useconditions
Boundary conditions of a thermal zone (usage, set temperatures, etc.)
buildingelement
Base class of all structural elements, calculation of parameters
wall
window
innerwall
outerwall
ceiling
floor
groundfloor
rooftop
archetype
Base class for all archetype buildings
residential
nonresidential
singlefamilydwelling
BMVBS defintion of a residential building
office
BMVBS defintion of an office building
inheritance
parent child relationship

Methodology: EnergyADE

- Using as many existing methods and classes in TEASER
- Creating new bindings for the current version of the EnergyADE
- Extracting all available parameters from EnergyADE
- Using Python Dictionaries similar to TEASERs data structure – GML-ID as keys
- Mapping GML attributes to corresponding TEASER building variables



```python
"""ThermalZone: BoundedBy/ThermalBoundaries ThermalOpenings"""

tzb_boundedby = gml_bind.featureMember[0].Feature.GenericApplicationPropertyOfAbstractBuilding[
    3].AbstractThermalZone.boundedBy_
tzb_dict = {}
tzb_dict_openings = {}
for i, bounded_object in enumerate(tzb_boundedby):
    tzb_type = bounded_object.ThermalBoundary.thermalBoundaryType
    tzb_gid = bounded_object.ThermalBoundary.id
    tzb_lcorner = bounded_object.ThermalBoundary.boundedBy.Envelope.lowerCorner.value()
    tzb_ucorner = bounded_object.ThermalBoundary.boundedBy.Envelope.upperCorner.value()
    tzb_azimuth = bounded_object.ThermalBoundary.azimuth.value()
    tzb_azimuth_uom = bounded_object.ThermalBoundary.azimuth.uom
    tzb_inclination = bounded_object.ThermalBoundary.inclination.value()
    tzb_inclination_uom = bounded_object.ThermalBoundary.inclination.uom
    tzb_area = bounded_object.ThermalBoundary.area.value()
    tzb_area_uom = bounded_object.ThermalBoundary.area.uom
    tzb_constr_href = bounded_object.ThermalBoundary.construction.href.strip('#')
    tzb_delimits_href = bounded_object.ThermalBoundary.delimits[0].href.strip('#')

    print(f'The {i+1}. thermal boundary is a {tzb_type} with id:{tzb_gid}:\n'
          f'Lower Corner:{tzb_lcorner}\nUpper Corner:{tzb_ucorner}\n'
          f'Azimuth: {tzb_azimuth} {tzb_azimuth_uom}\n'
          f'Inclination: {tzb_inclination} {tzb_inclination_uom}\n'
          f'Area: {tzb_area} {tzb_area_uom}\n'
          f'Construction href:{tzb_constr_href}\n'
          f'Delimits href: {tzb_delimits_href}')

    tzb_dict.update({tzb_gid: [tzb_type, tzb_azimuth, tzb_inclination, tzb_area, tzb_constr_href]})

    try:
        for openings in bounded_object.ThermalBoundary.contains:
            tzb_contains_tho_id = openings.ThermalOpening.id
            tzb_contains_tho_area = openings.ThermalOpening.area.value()
            tzb_contains_tho_area_uom = openings.ThermalOpening.area.uom
            tzb_contains_tho_contr_href = openings.ThermalOpening. \
                construction.href.strip('#')

            print(f'Thermal Opening(Window/Door): {tzb_contains_tho_id}\n'
                  f'Area : {tzb_contains_tho_area} {tzb_contains_tho_area_uom}\n')

            tzb_dict_openings.update({tzb_contains_tho_id: [tzb_gid, tzb_contains_tho_area, tzb_inclination,
                                                            tzb_azimuth, tzb_contains_tho_contr_href]})
    except IndexError:
        continue
"""trying to set the Outer Wall / with Layers/ Materials"""

for key, value in tzb_dict.items():
    if value[0] == "outerWall":
        out_wall = OuterWall(parent=tz)
        out_wall.name = key
        for key_openings, value_openings in tzb_dict_openings.items():
            if key == key_openings:
                out_wall.area = value[3]-value_openings[1]
                break
            else:
                out_wall.area = value[3]
        out_wall.orientation = value[1]
        out_wall.tilt = value[2]
        for layers in layer_dict[value[4]]:
            layer = Layer(parent=roof, id=layers[0])
            layer.thickness = layers[2]
            material = Material(parent=layer)
            material.name = material_dict[layers[3]][0][0]
            if material_dict[layers[3]][0][0] == 'Luftschicht':
                rvalue = material_dict[layers[3]][0][2]
                material.thermal_conduc = 0.02225
                material.density = 1.2041
                material.heat_capac = 1
            else:
                material.density = material_dict[layers[3]][0][1]
                material.heat_capac = material_dict[layers[3]][0][2]
                material.thermal_conduc = material_dict[layers[3]][0][3]
            BuildingElement.add_layer(out_wall, layer=layer)

        # out_wall.load_type_element(year=bldg.year_of_construction, construction='heavy')
```

Example Code: Mapping EnergyADE Attributes to corresponding variables in TEASER

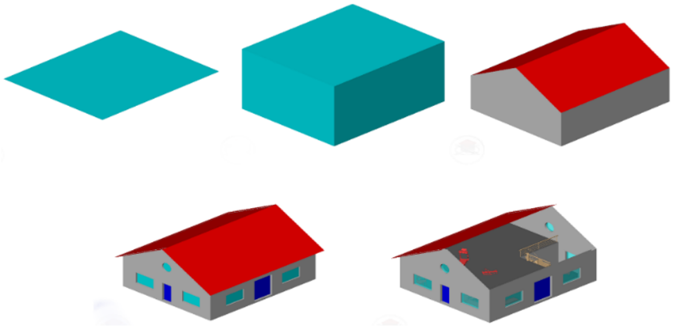
Workflow

Necessary Requirements:

- Python IDE
- TEASER+ (Now based on TEASER v0.66)
- TEASER dependencies (Mako, PyXB, SciPy…)
- BuildingsPy / Python-Dymola Interface
- Dymola (2018)
- AixLib (v0.7.3)
- Matplotlib for visualization

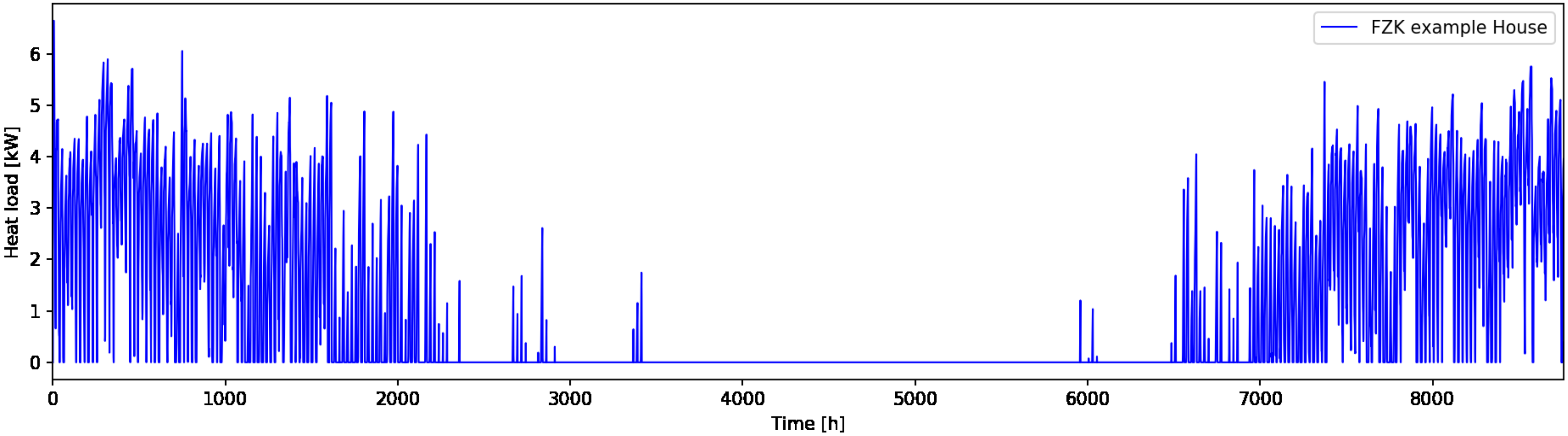
Workflow was tested on the FZK House

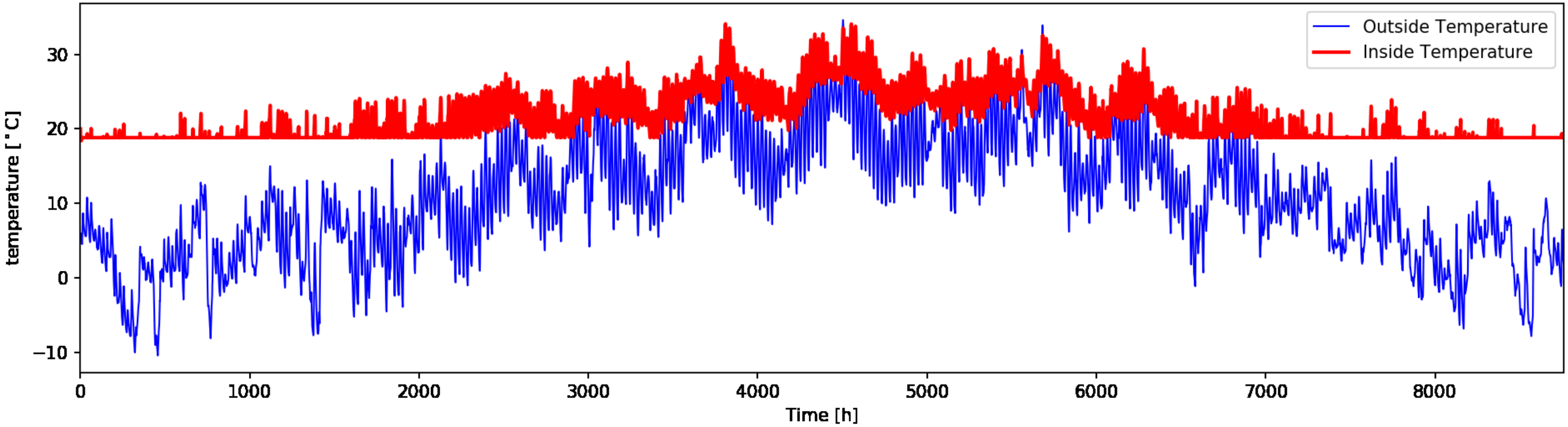
And could look something like that!



Test Subject : FZK-House (2017): FZK Haus - CityGML Wiki. http://www.citygmlwiki.org/index.php?title=FZK_Haus

Results


FZK example House
Heat load [kW]
0
1
2
3
4
5
6
0
1000
2000
3000
4000
5000
6000
7000
8000
Time [h]


Outside Temperature
Inside Temperature
temperature [$^\circ$ C]
−10
0
10
20
30
0
1000
2000
3000
4000
5000
6000
7000
8000
Time [h]

Discussion

- Fast computational times for annual heating loads, **BUT** inheriting some limitations of the Reduced Order Model:
  - Merging Walls leads to disregard of the orientation
  - Issues in fast transient situations
  - Issues in extreme building construction scenarios

  ↳ sometimes leading to unrealistic heat loads in the summer months or over/under estimations of the inside temperatures.

- Possible solutions and improvements:
  - Increase the Element Order (separate roof and base elements)
  - Adding RC-links in series to the existing wall elements
  - Changing the design (excitation) frequency
- Depending on the use-case and subsequent required time-resolution of the simulation

Future Work and Outlook

- Compare results with other simulation tools
- Test workflow with multiple Buildings (small District) → EnergyADE data needed!
- Introducing more precise volume calculations for smaller buildings without flat roofs
- Complete LoD4 data input integration
- Expand some internal function/methods for non-residential buildings
- Users decision for enrichment database / templates
- Extending the functionalities with “Cooling”
- Refine CityGML output (including EnergyADE)
- Release / Merge TEASER+ v0.1