Optimization of a complex energy supply system using PyFMI

EnOB
Forschung für Energieoptimiertes Bauen

Bundesministerium für Wirtschaft und Energie

Gesa Böhme

Fraunhofer-Institute for Solar Energy Systems ISE, Freiburg

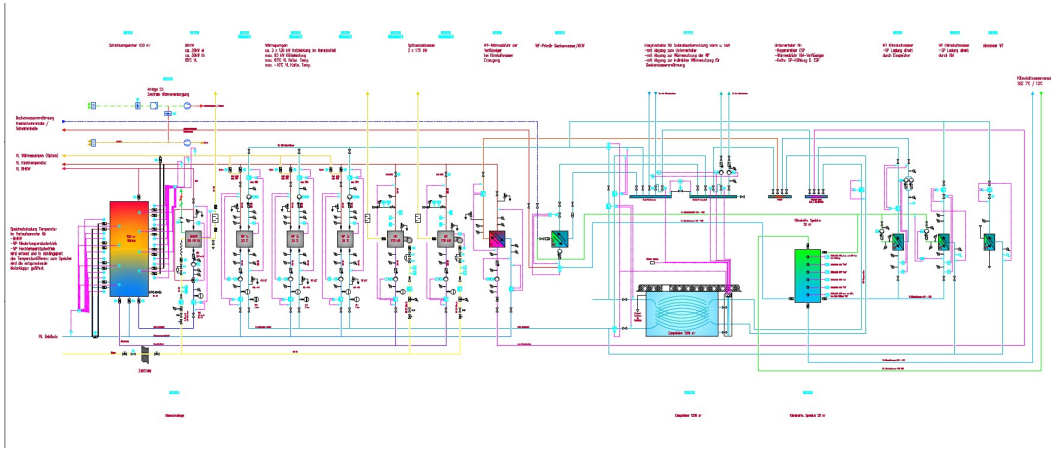
www.ise.fraunhofer.de

Thermal Systems Workshop

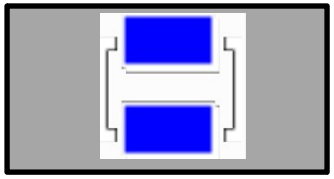
Freiburg, 23.03.2015

Fraunhofer ISE

Motivation



simulation model



Planning phase:

- concept selection
- viability check
- dimensioning

Operation phase:

- fault detection
- optimization

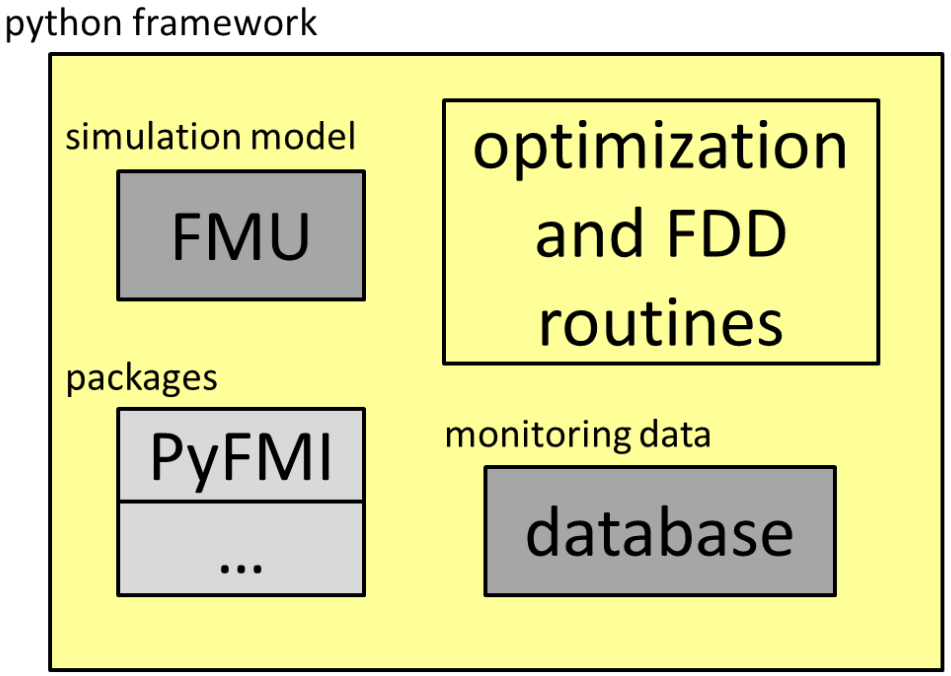
Framework

Procedure:

1. Export Simulation Model as FMU
2. Import FMU in Python via PyFMI

Advantage:

- open source
- license-independent
- access monitoring data



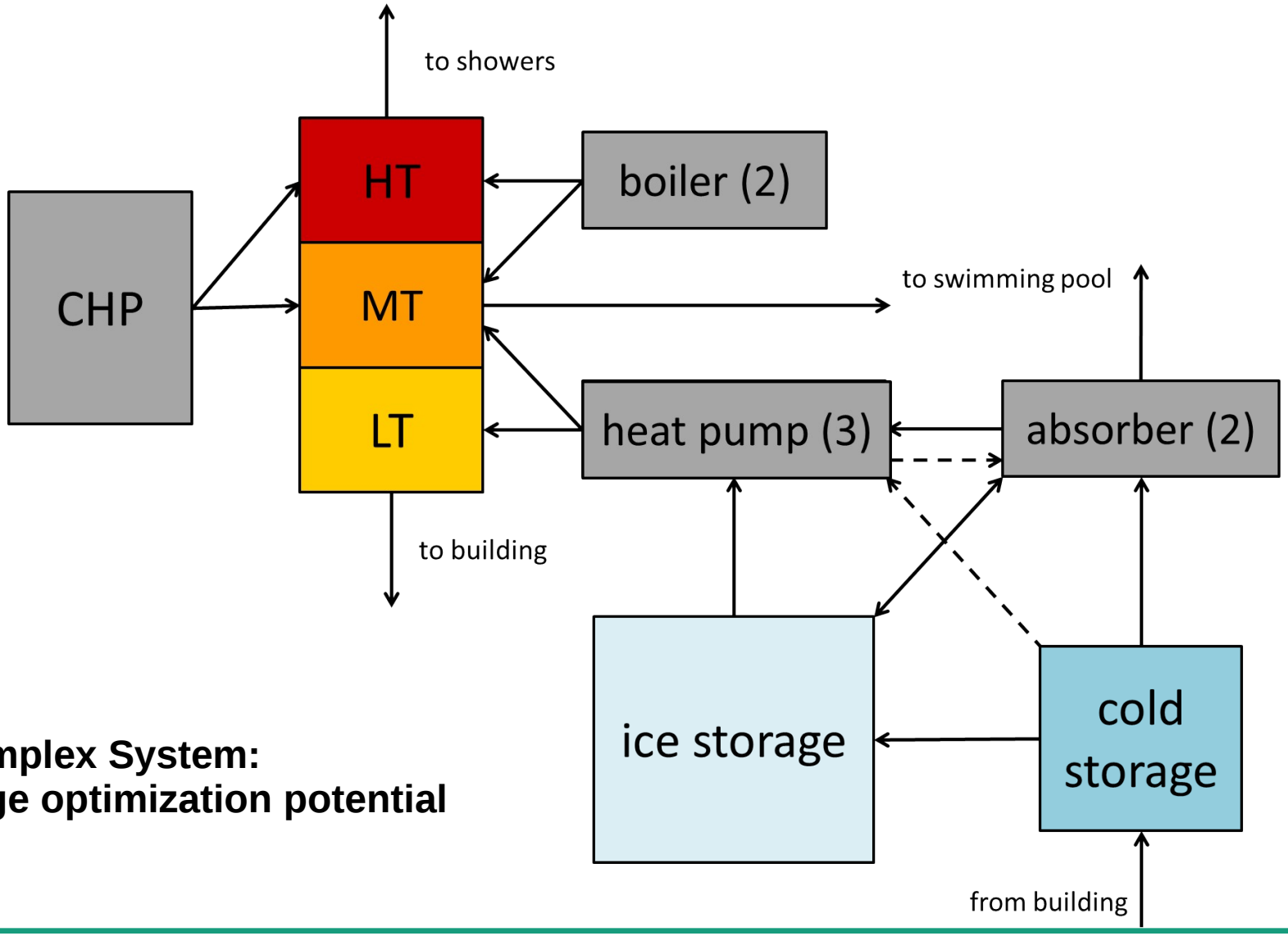
Application – 1) Technical Aspect

```
import pyfmi                                                    # import package PyFMI
model = pyfmi.load_fmu("example_model.fmu")                     # load model as FMU
stop_t = 24*60*60                                               # simulation settings
step_size = 60
model.set(["Par1","Input1"],[2010, 0.001])                      # set start values
model.initialize()                                              # initialize model

# --------------start simulation loop-----------------------------
i = 0
while i < stop_t :
    status = model.do_step(i, step_size)                        # perform simulation step
    if status == pyfmi.fmi.FMI_OK:                              # check simulation step
        act_T = model.get("iceStorage.T")                       # get output value
        v_new = Opti(act_T, args)                               # OPTIMIZATION
        model.set("volumeFlow_iceStorage", v_new)               # set input value
        i = i + step_size
    else:
        print "error"
        break
# --------------end simulation loop-------------------------------
```

Application – 1) Technical Aspect

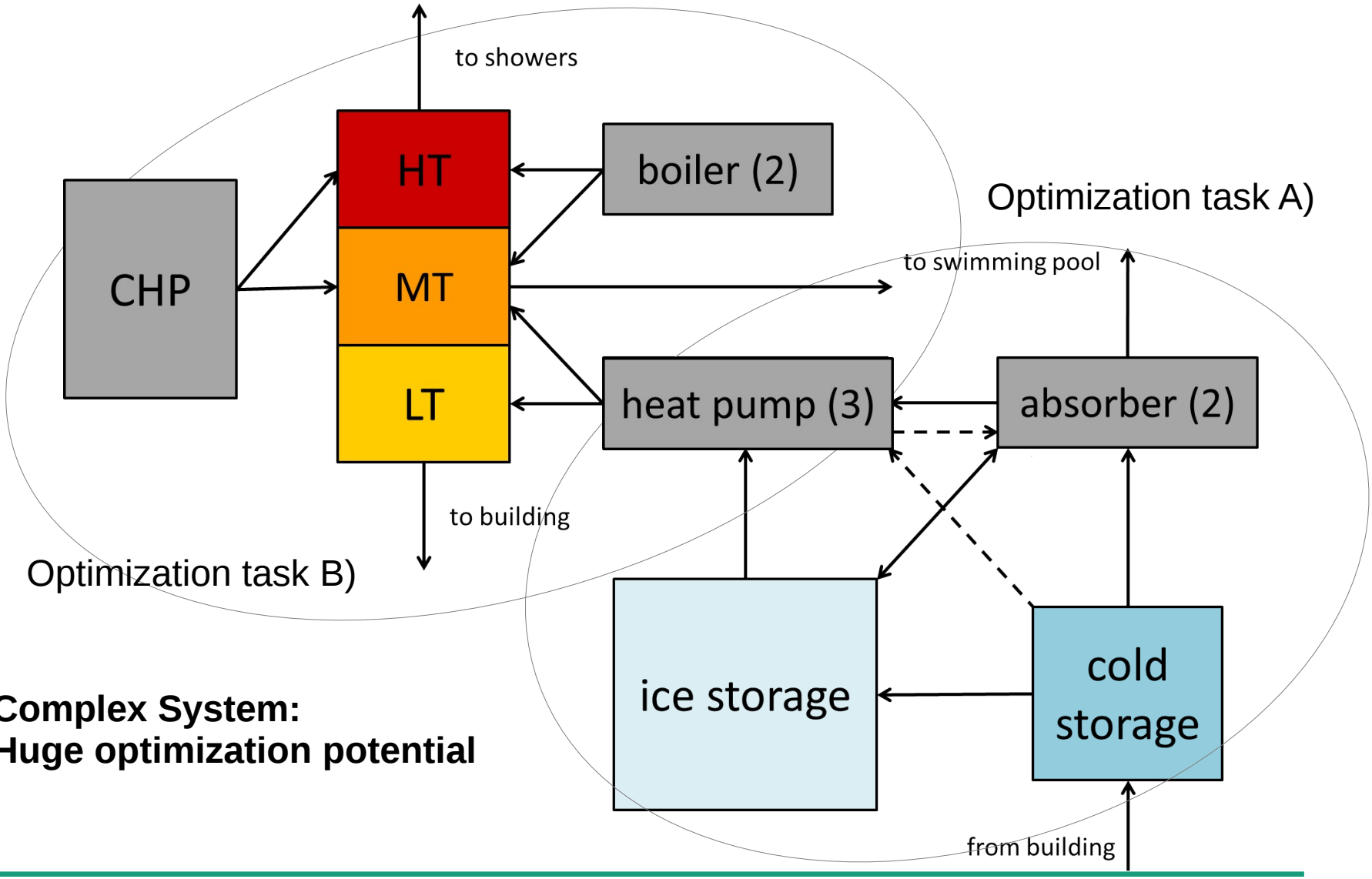
```
import pyfmi                                                    # import package PyFMI
model = pyfmi.load_fmu("example_model.fmu")                     # load model as FMU
stop_t = 24*60*60                                               # simulation settings
step_size = 60
model.set(["Par1","Input1"],[2010, 0.001])                      # set start values
model.initialize()                                              # initialize model

# --------------start simulation loop-----------------------------
i = 0
while i < stop_t :
    status = model.do_step(i, step_size)                        # perform simulation step
    if status == pyfmi.fmi.FMI_OK:                              # check simulation step
        act_T = model.get("iceStorage.T")                       # get output value
        v_new = Opti(act_T, args)                               # OPTIMIZATION
        model.set("volumeFlow_iceStorage", v_new)               # set input value
        i = i + step_size
    else:
        print "error"
        break
# --------------end simulation loop-------------------------------
```

Application – 2) Physical Aspect



Complex System:
Huge optimization potential

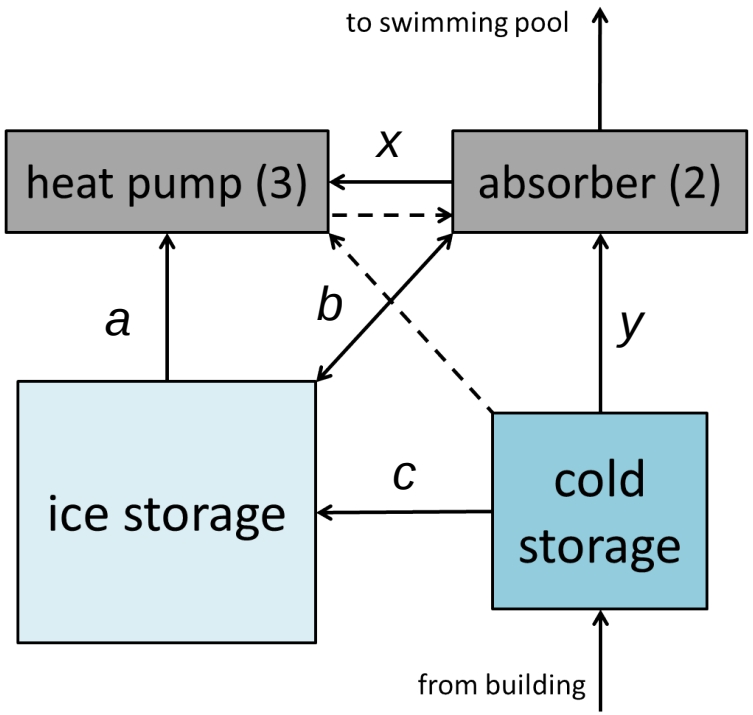
Application – 2) Physical Aspect



Complex System:
Huge optimization potential

Application – 2) Physical Aspect

Optimization Task A



optimization parameters:

- volume flows *a, b, c, x ,y*

optimization objectives:

a) high COP → *f(a,x)*

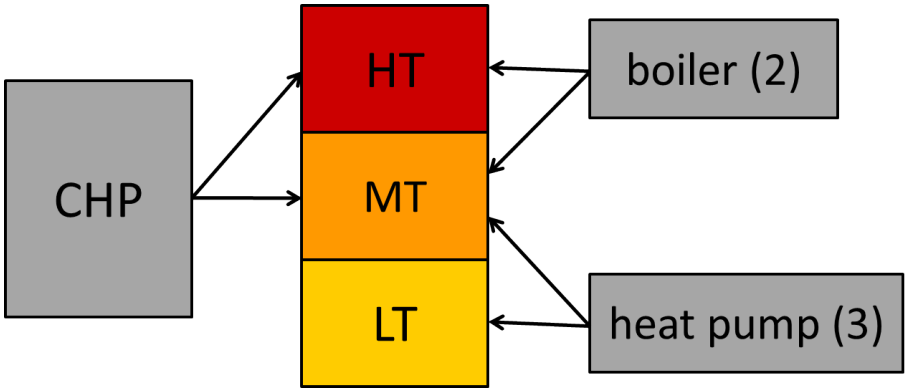
b) optimal ice storage temperature → *g(a,b,c)*

c) optimal cold storage temperature → *h(c,y)*

- formulation of common objective function
- prioritization

Application – 2) Physical Aspect

Optimization Task B



HT-demand: v(boiler, CHP)
MT-demand: y(boiler, CHP, heat pumps)
NT-demand: η(heat pumps)

optimization parameters:

- set values for heat generators (discrete)

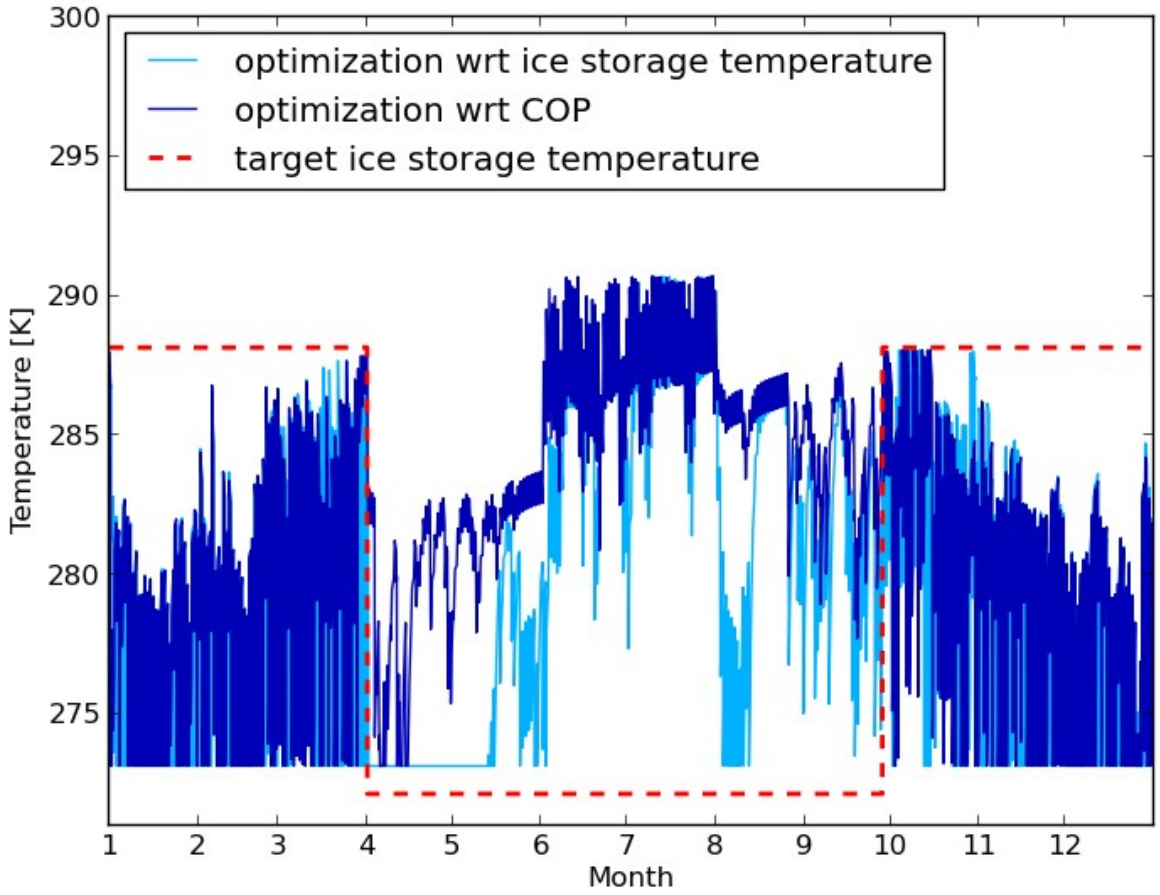
optimization objectives:

a) low energy costs
b) low CO_2 emissions

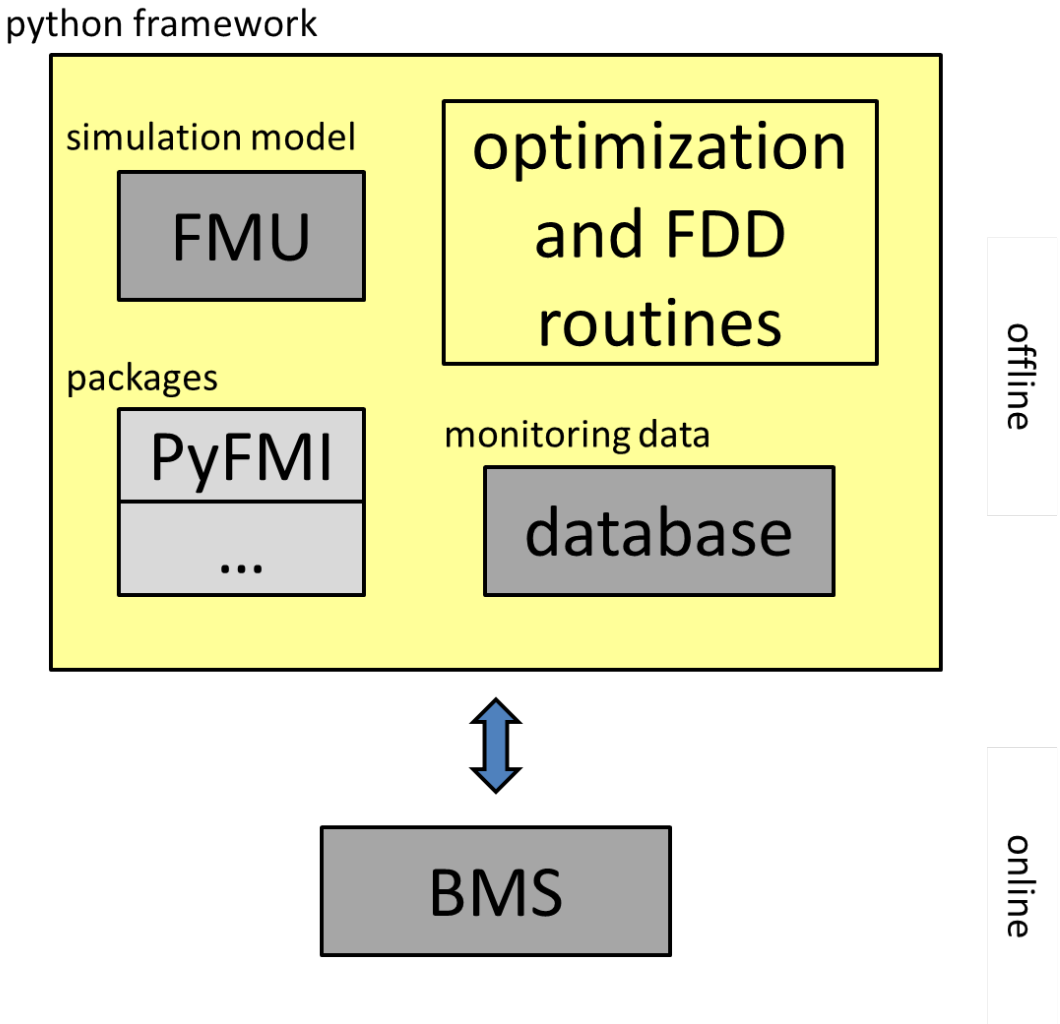
bounds:

- demand coverage

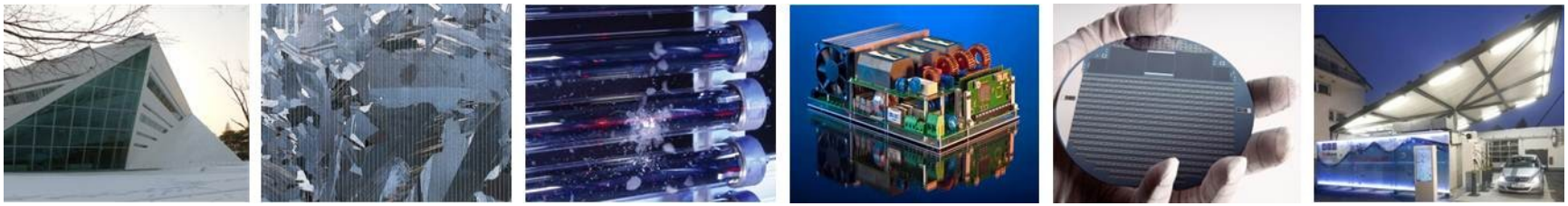
Application – 3) Results



Outlook



Thank you for your attention!



Fraunhofer-Institute for Solar Energy Systems ISE

Gesa Böhme

www.ise.fraunhofer.de
gesa.boehme@ise.fraunhofer.de