ETH zürich

Research Center for
Energy Networks FEN

A framework for the simulation of distributed energy systems

28.09.2021

Dr. Adamantios Marinakis
Principal Expert
Research Center for Energy Networks, ETHZ

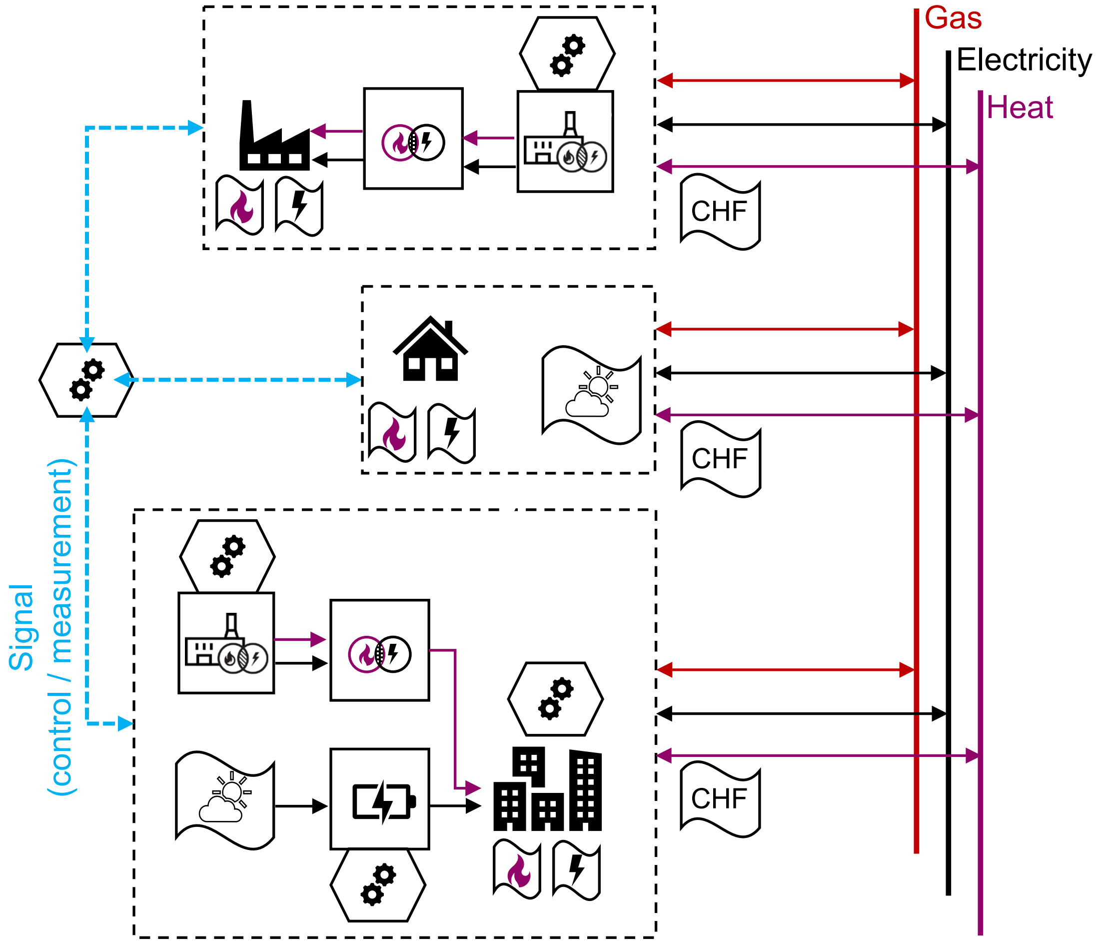
Agenda

1. What is the ReMaP Simulation Framework (SFW)
2. Features of the SFW
3. Basic software design of the SFW
4. Reference to already implemented “experiments” (use cases)

Agenda

1. **What is the ReMaP Simulation Framework (SFW)**
2. Features of the SFW
3. Basic software design of the SFW
4. Reference to already implemented “experiments” (use cases)

ReMaP Simulation Framework (SFW)
Representing a multi-energy-carrier district from utility to site level

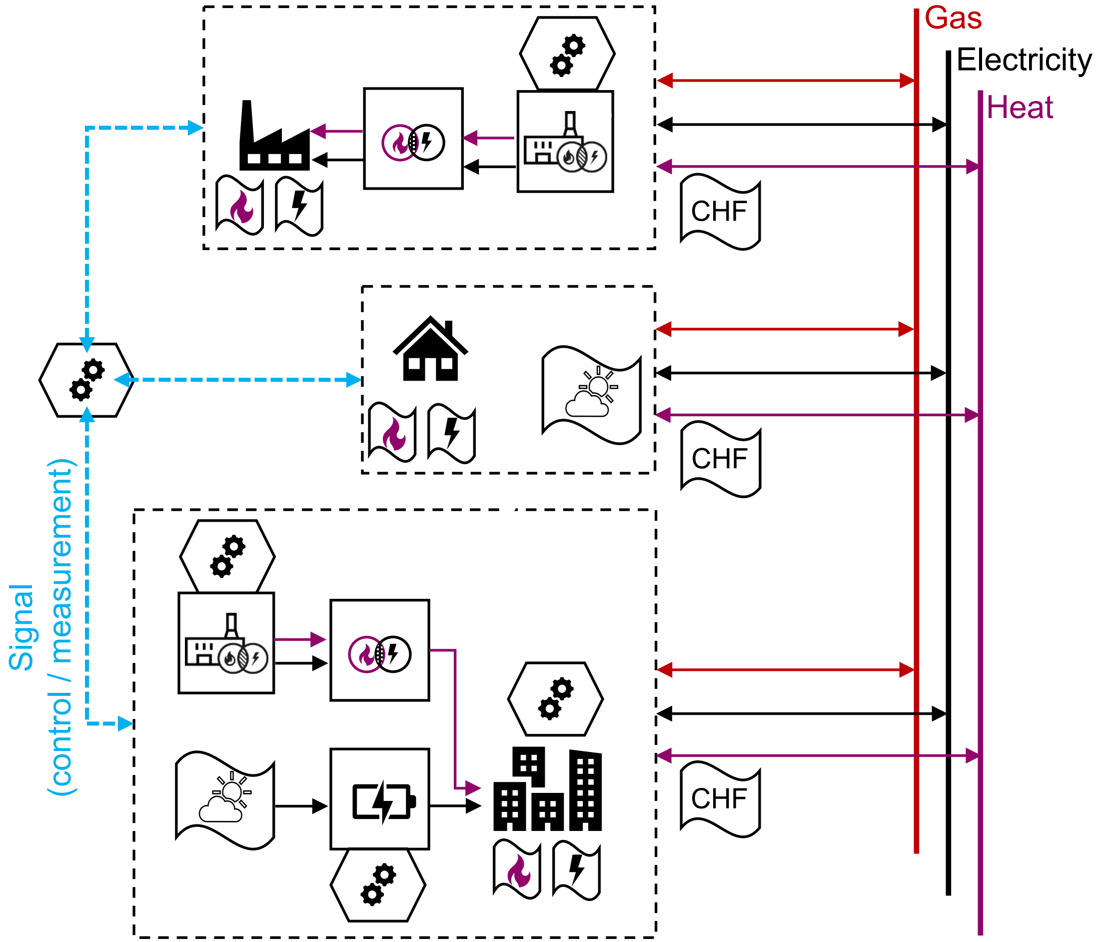


What

- A software framework allowing to simulate the operation of all energy sectors in a district
- Considers utility-level and site-level components
- Allows for simultaneous representation of a plurality of operational logic / controllers
- Time step and simulation horizon selected by the user depending on the considered case
- Can be used as part of a hardware-in-the-loop simulation

Heat pump
CHP
Battery
Solar PV
Heat demand
Electricity demand
Controller
CHF Energy price

ReMaP Simulation Framework (SFW)
Representing a multi-energy-carrier district from utility to site level



What

- A software framework allowing to simulate the operation of all energy sectors in a district
- Considers utility-level and site-level components
- Allows for simultaneous representation of a plurality of operational logic / controllers
- Time step and simulation horizon selected by the user depending on the considered case
- Can be used as part of a hardware-in-the-loop simulation

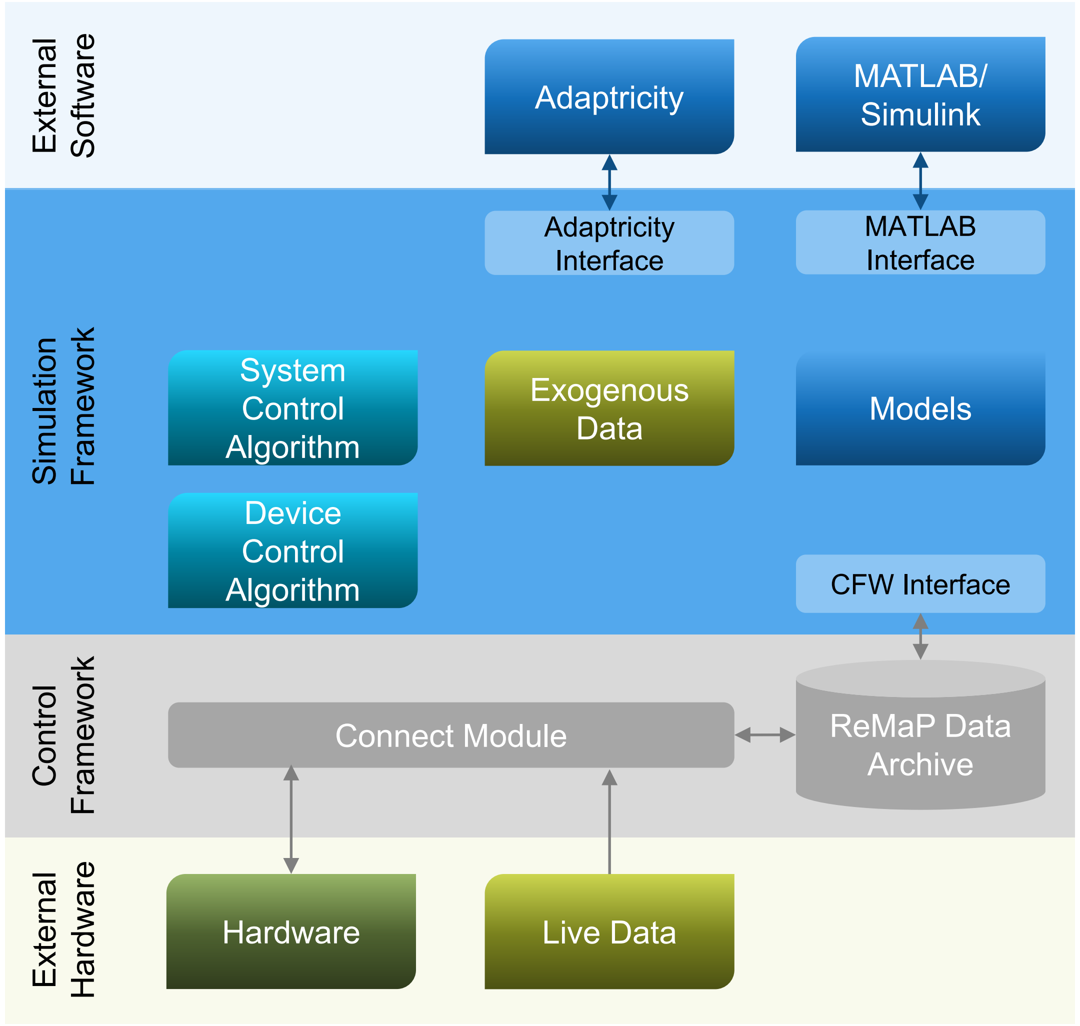
Attributes

- Open-source python-based software
- Model library with independent hardware models and control algorithms
- Modular design; user can add his/her own models or algorithms
- Can connect to external software

Agenda

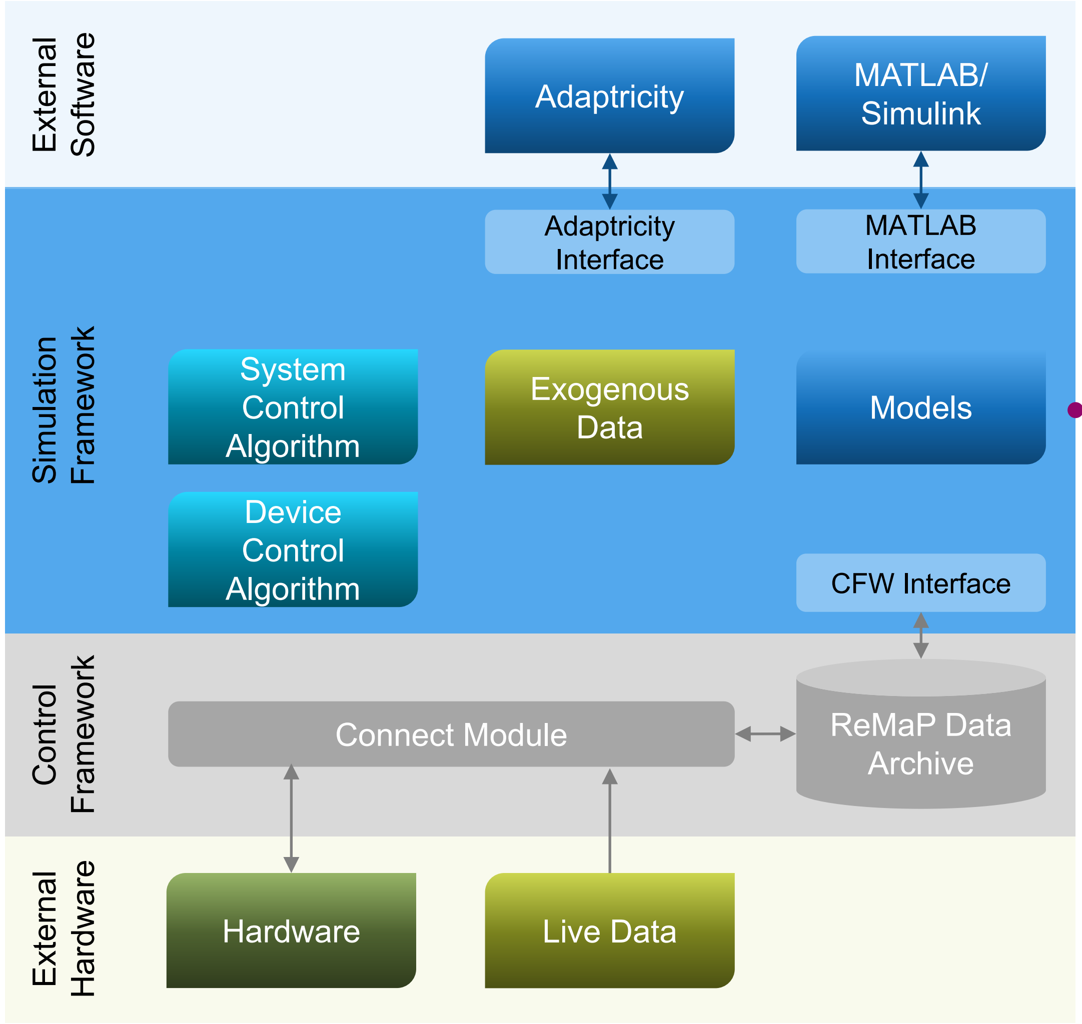
1. What is the ReMaP Simulation Framework (SFW)
2. **Features of the SFW**
3. Basic software design of the SFW
4. Reference to already implemented “experiments” (use cases)

Features & Interconnections



Data Controller Model Hardware

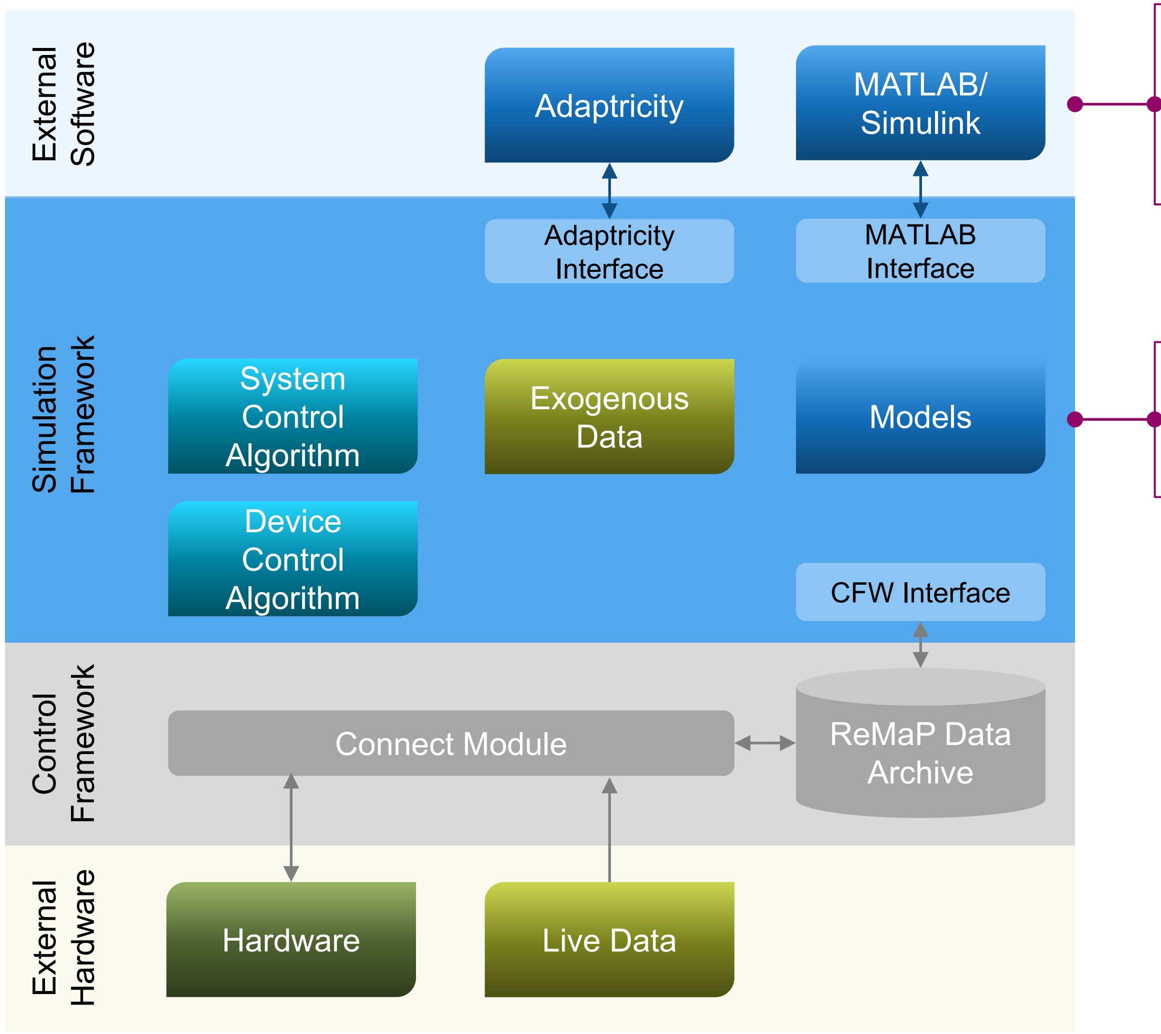
Features & Interconnections



- ✓ Plug-and-play-ability for different independent models, algorithms and data to design systems per the user's needs
- ✓ No coding required by the user

Data | Controller | Model | Hardware

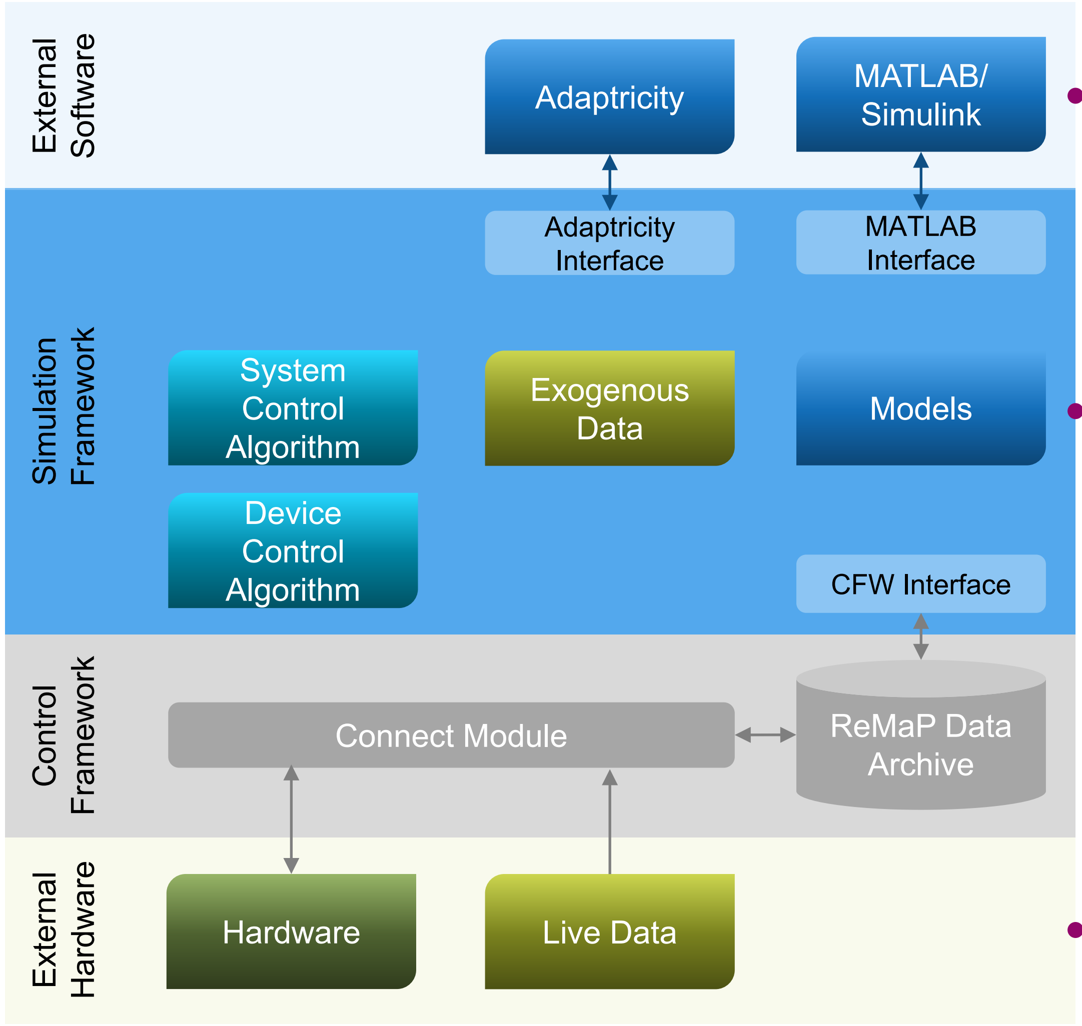
Features & Interconnections



- ✓ Ability to connect to external software, where components can be represented and smoothly integrated into the simulation
- ✓ Appropriate interfaces are developed for selected cases

- ✓ Plug-and-play-ability for different independent models, algorithms and data to design systems per the user's needs
- ✓ No coding required by the user

Features & Interconnections



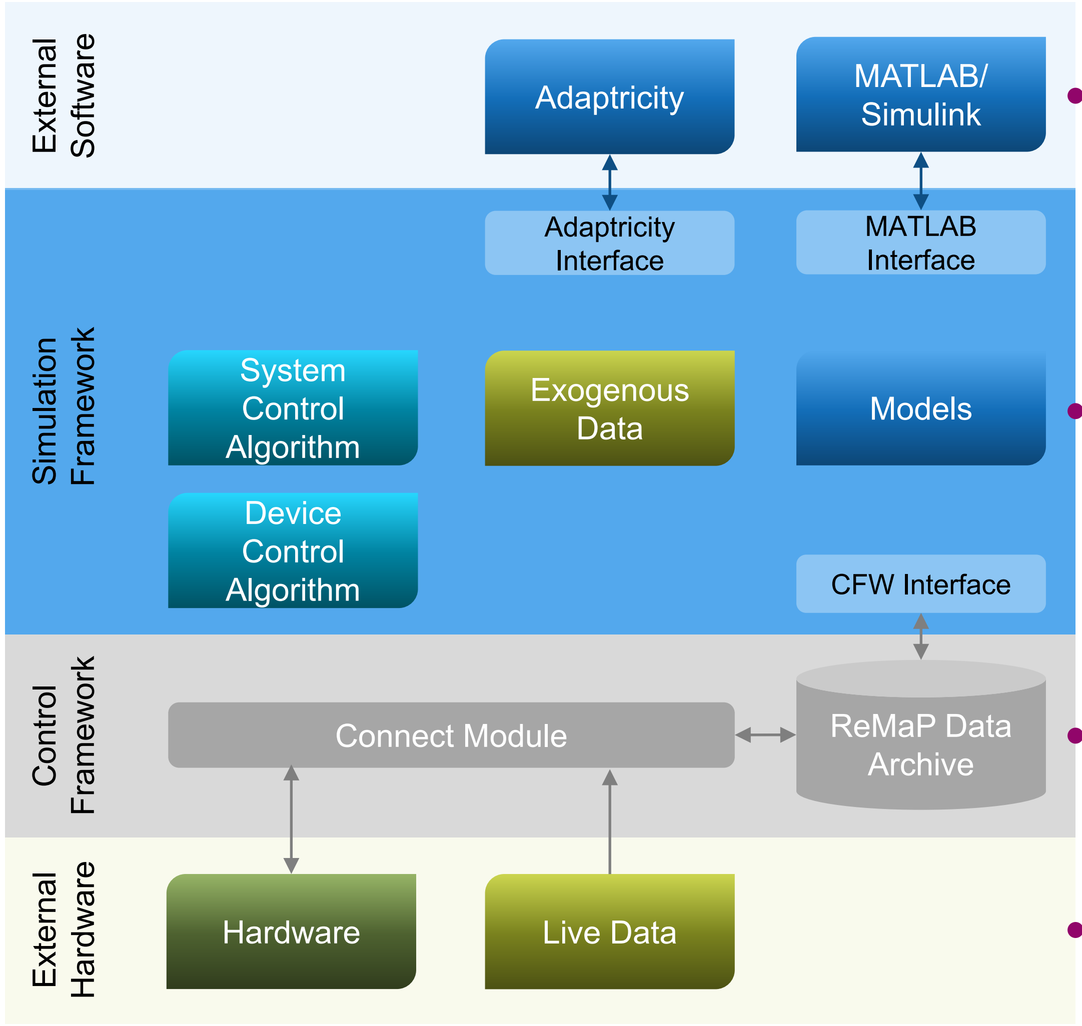
- ✓ Ability to connect to external software, where components can be represented and smoothly integrated into the simulation
- ✓ Appropriate interfaces are developed for selected cases

- ✓ Plug-and-play-ability for different independent models, algorithms and data to design systems per the user's needs
- ✓ No coding required by the user

- ✓ Ability to connect to physical hardware components and to sources of live data (e.g. PV production), located at different places

Data | Controller | Model | Hardware

Features & Interconnections



- ✓ Ability to connect to external software, where components can be represented and smoothly integrated into the simulation
- ✓ Appropriate interfaces are developed for selected cases

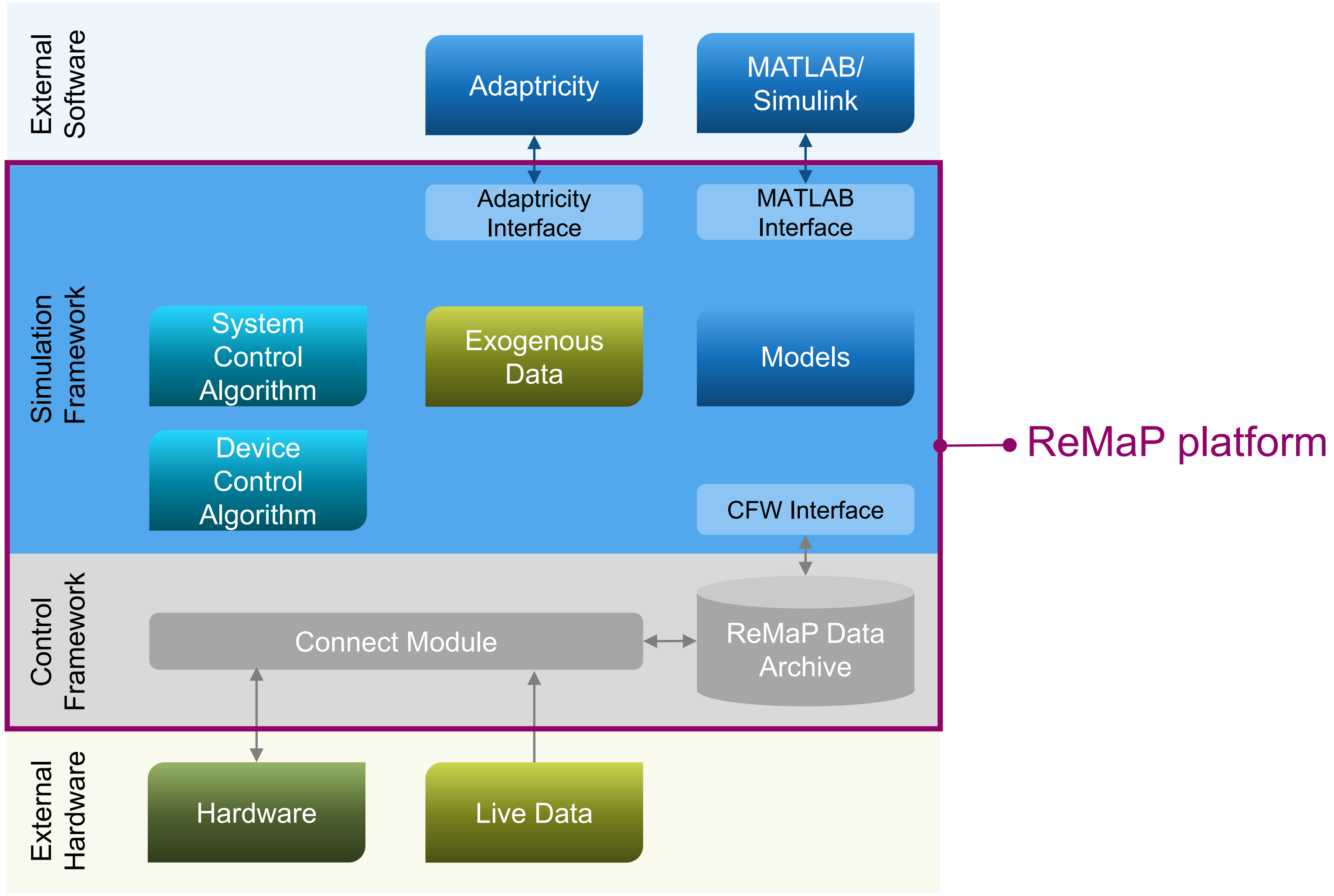
- ✓ Plug-and-play-ability for different independent models, algorithms and data to design systems per the user's needs
- ✓ No coding required by the user

- ✓ Interfaced to the ReMaP Control Framework (CFW), allowing
  1. access to the Data Archive and
  2. communication with the physical hardware

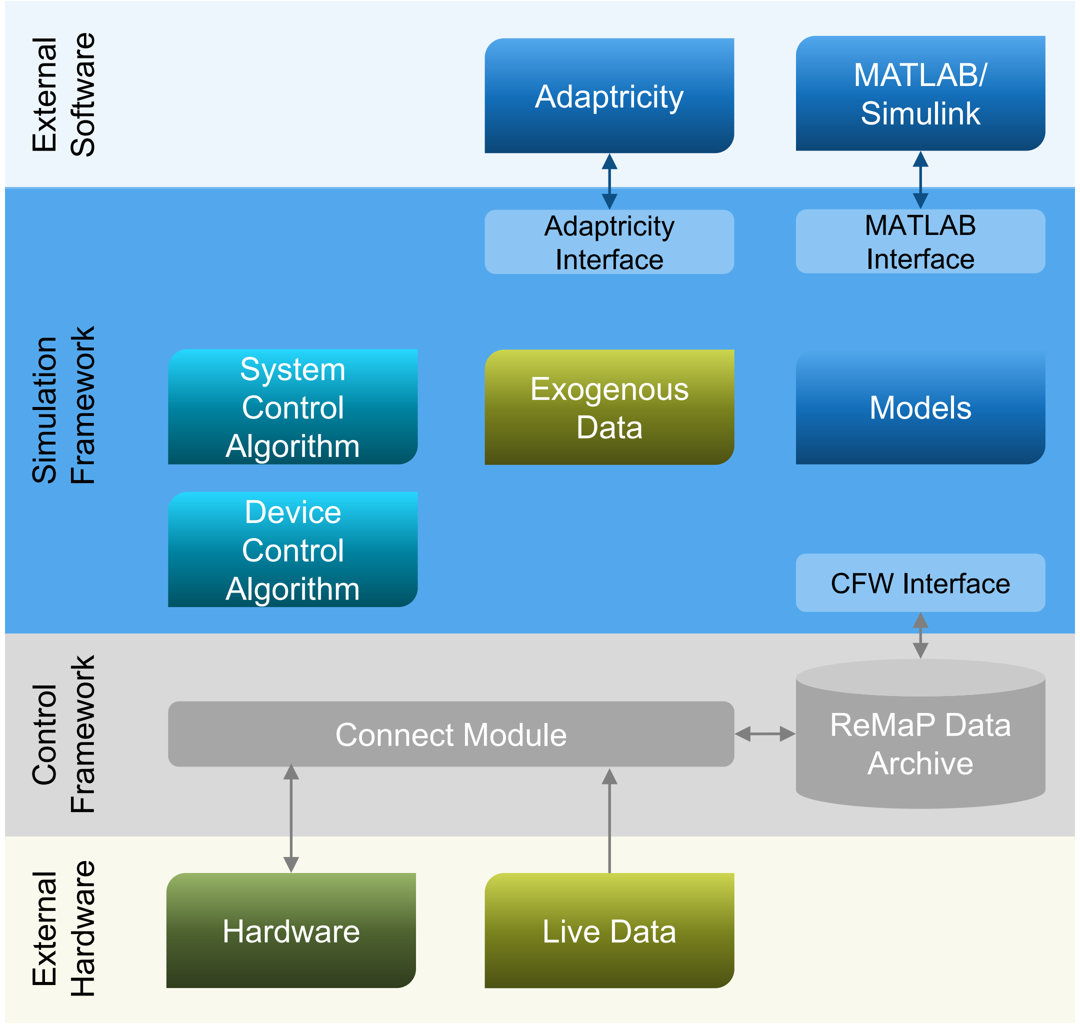
- ✓ Ability to connect to physical hardware components and to sources of live data (e.g. PV production), located at different places

Data Controller Model Hardware

Features & Interconnections



Model Library (continuously expanded)



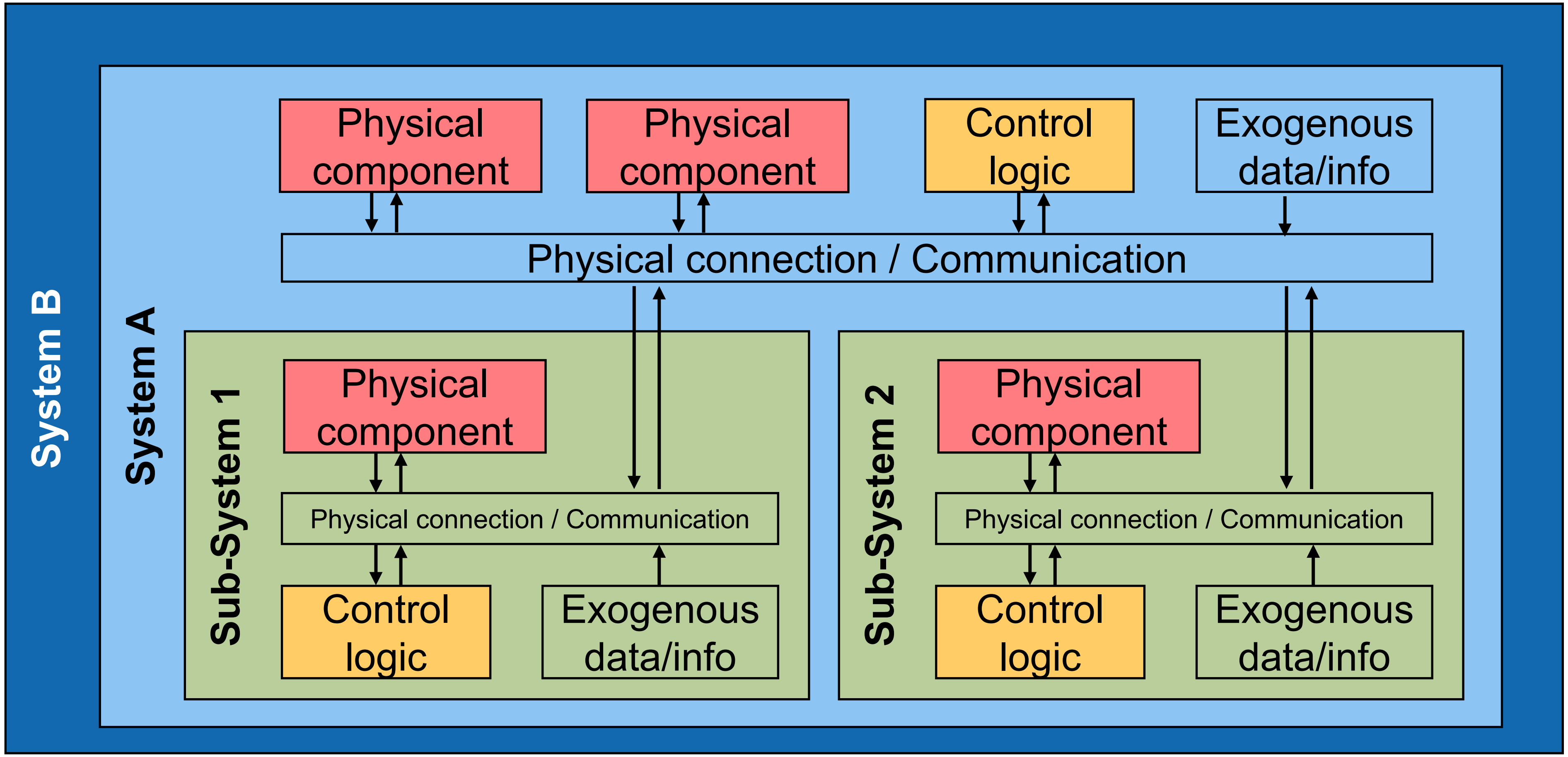
Models integrated up to now

- Electricity Network
- Battery Storage
- Combined Heat and Power (CHP)
- Electrolyser
- Generation Timeseries (Wind, PV, etc.)
- Load Timeseries (Electric, Heat, Gas)
- Fuel Cell
- Heat Pump
- Hydrogen Storage
- Methanation Reactor
- Thermal Energy Storage (TES)

Agenda

1. What is the ReMaP Simulation Framework (SFW)
2. Features of the SFW
3. **Basic software design of the SFW**
4. Reference to already implemented “experiments” (use cases)

Basic architecture of SFW: A hierarchy of systems



Systems:

- pre-defined
- user-defined

Physical component:

- computeState()

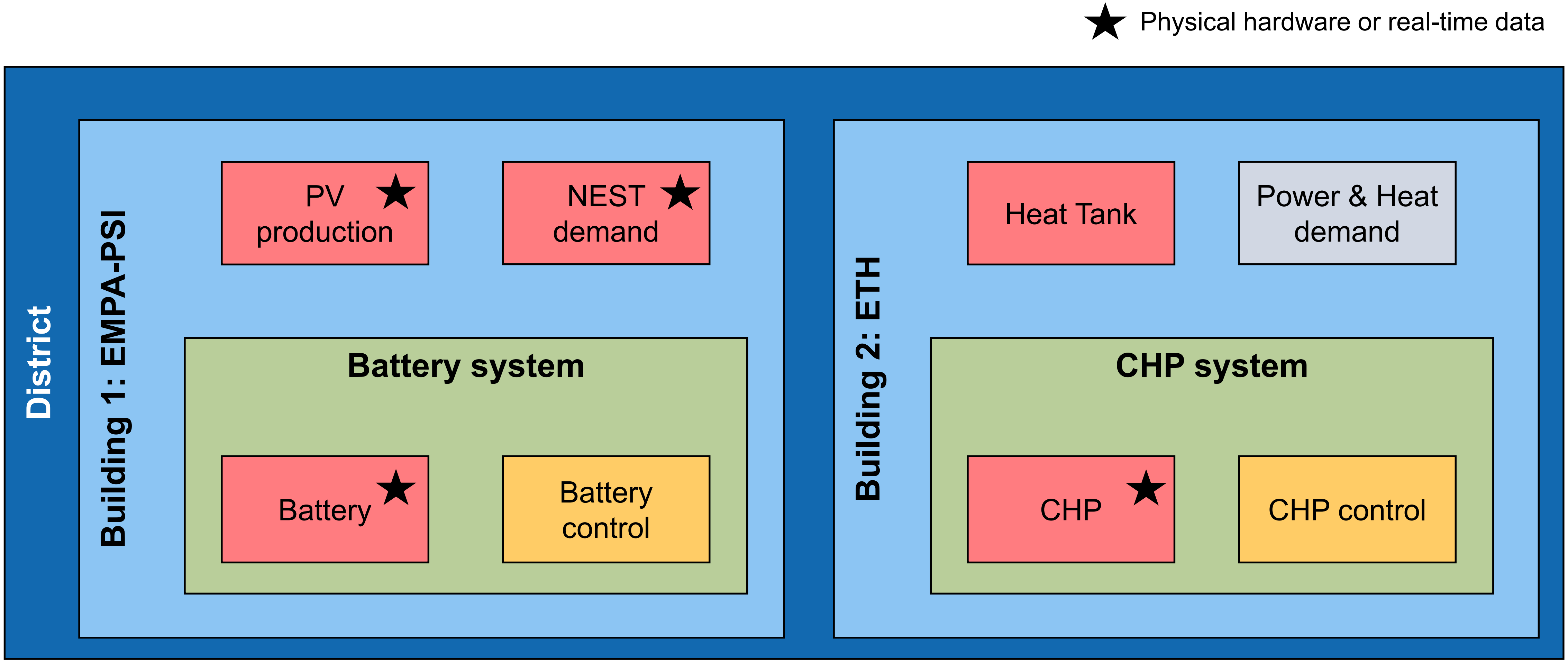
Control logic:

- computeAction()

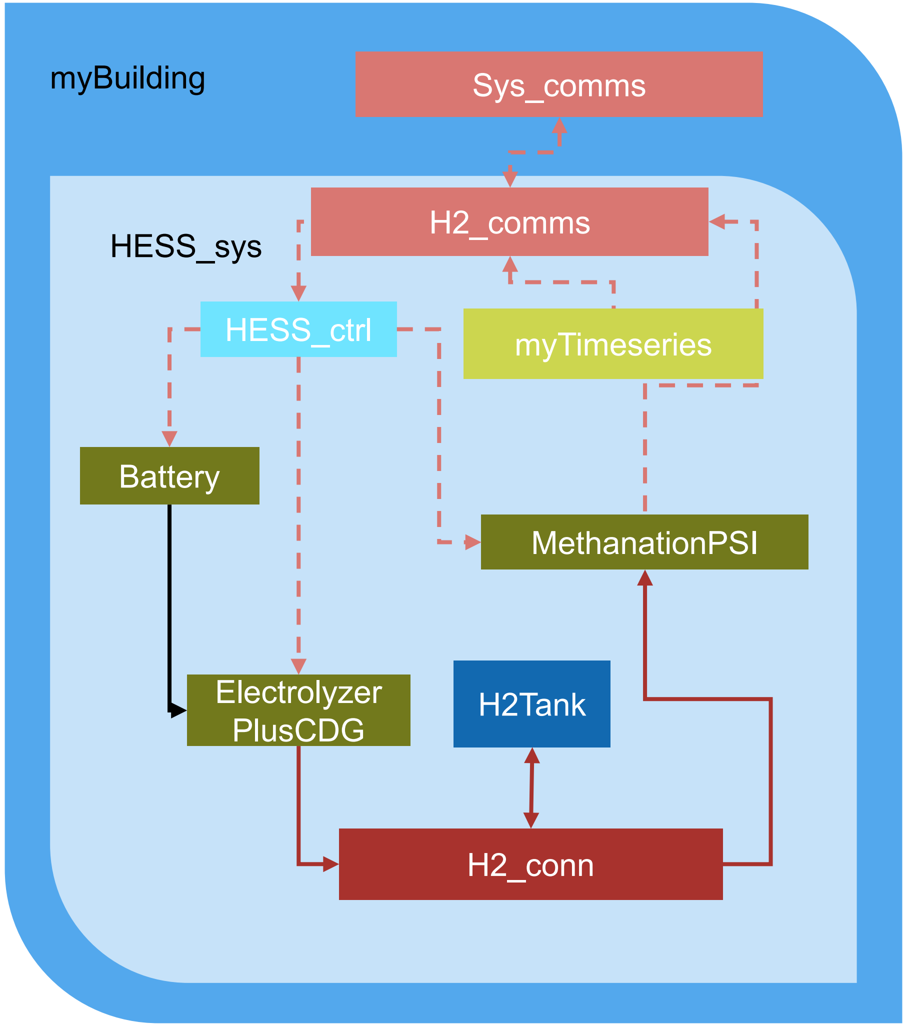
Exogenous data/info:

- timeseries/logic

Basic architecture of SFW: A hierarchy of systems



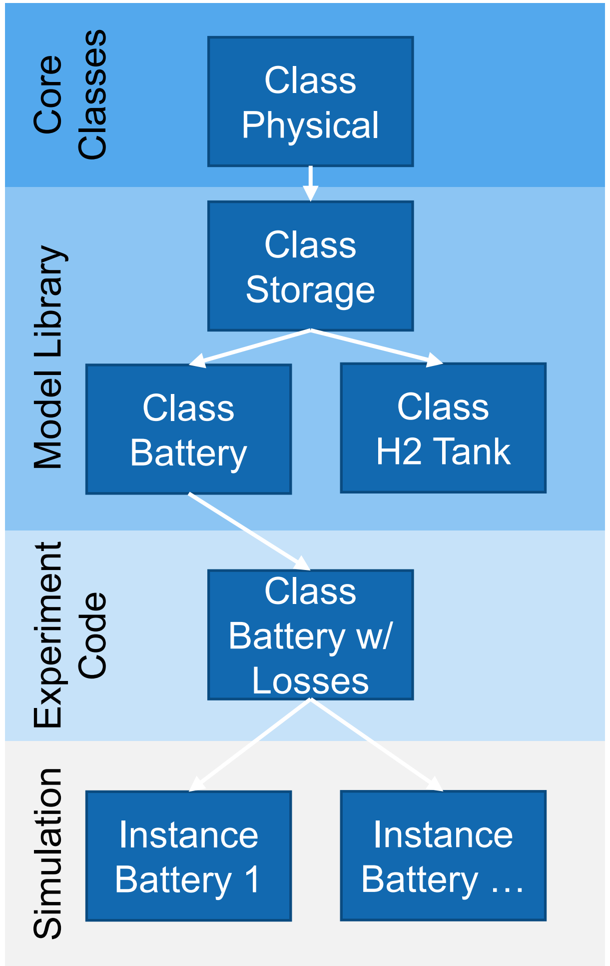
User Interface




```
myTimeseries = {'Name': ExogenousElectricity, 'Data': el_production}

# Hydrogen Energy Storage System setup
HESS_phys = {'Name': [Battery, ElectrolyserPlusCDG, MethanationPSI, H2Tank],
        'Param': [[{'P_max': float('inf'), 'P_min': -float('inf'), 'max_cap': Bat_cap, 'lookback’:
                    window_length, 'factor_battery': factor_bat},
                   {'P_init': P_bat_init, 'SoC_init': SoC_bat_init}, {'factor_time': factor_time}],
                  [{'eff_HHV': eta_ely, 'eta_cdg': eta_cdg, 'rated_input': float('inf')},
                   {'P_init': P_ely_init}, {'P_el_key': 'P_Ely_setpoint', 'leakage_offset': leakage_offset}],
                  [{'P_H2_rated': MR_rating, 'factor_methanation': factor_methanation, 'eff': MR_eff},
                   {'H2_init': MR_H2_init}, {}],
                  [{'capacity': float('inf')}, {'SoC_init': H2_SoC_init},
              {'P_net_key': 'H2_conn_up_00', 'sim_H2_tank': sim_H2_tank, 'factor_time': factor_time}]]}
H2_control = {'Name': HESS_ctrl,
        'Param': [{'factor_time': factor_time, 'factor_ely': factor_ely, 'factor_bat': factor_bat, 'llim': llim,
              'ulim': ulim, 'cap_H2': H2_tank_cap, 'cap_bat': Bat_cap, 'eta_ely': eta_ely,
              'SoC_bat_lim_low': SoC_bat_lim_low, 'eta_elysim': eta_elysim, 'eta_cdg': eta_cdg,
              'P_Ely_max': P_ely_max, 'leakage_offset': leakage_offset, 'P_Bat_max': P_bat_max,
              'P_Bat_min': P_bat_min, 'SoC_min': Min_SoC, 'SoC_max': Max_SoC},
             {'u_Ely_init': ctrlEly}, {}]}
H2_conn = {'Name': [H2Connector], 'Param': [[{'outputConnections': 1}, {'name': 'H2_conn'}, {}]]}
H2_comms = {'Name': [HESS_comms], 'Param': [{}]}

# Overall system setup
sysObjs = {'Name': [HESS_sys], 'Param': [[H2_control, myTimeseries, HESS_phys, H2_conn, H2_comms]]}
sys_comms = {'Name': [SysComms], 'Param': [{}]}

myBuilding = Building(None, None, None, sysObjs, None, sys_comms)
```

- Timeseries Data
- Execution Order
- Battery Param.
- Electrolyser Param.
- Methane Param.
- H2 Tank Param.
- Control Param.
- Connector
- HESS Comms
- System Layout
- Building Comms
- Overall Model

Object-oriented approach




```
class Storage(coreClasses.Physical):
"""Base class for all storage systems: Storage class inherits from Component class
    and implements basic storage with physical limits"""
    def __init__(self, model_param, init_param, sim_param):
        super().__init__(model_param, init_param, sim_param)
```

Storage parameters

```
        self.P_net = init_param.get('P_init', 0) # Net power flow into storage in W
        self.SoC = init_param.get('SoC_init', 0) # State of charge in Wh
        self.max_cap = model_param.get('capacity', float('inf')) # Max capacity in Wh
        self.P_net_key = sim_param.get('P_net_key', 'P_STRG_in') # Key name for P_net input
```

Base method calling storage model code

```
    def computeState(self, com_bus, connector):
        self.P_net = connector[self.storage_type].bus['connected_up'][self.P_net_key].power
        self.run_storage_model()
        return [self.SoC, self.P_net]
```

Actual model formulation

```
    def run_storage_model(self):
        self.SoC = self.SoC + self.P_net * TimeKeeping.dt / 3600
        self.SoC = self.check_physical_lims(self.SoC, 0, self.max_cap, action='lim&warn’)
```

```
class BatteryLosses(Battery):
```

Extend model parameters

```
    def __init__(self, model_param, init_param, sim_param):
        super().__init__(model_param, init_param, sim_param)
        self.charge_loss = model_param.get('charge_loss', 0.01) # Charging loss in percent
        self.discharge_loss = model_param.get('discharge_loss', 0.01) # Discharging loss in %
```

Add experiment specific modelling detail

```
    def account_for_losses(self):
        if self.P_net > 0:
            self.P_net *= (1 - self.charge_loss)
        elif self.P_net < 0:
            self.P_net *= 1 / (1 - self.discharge_loss)
```

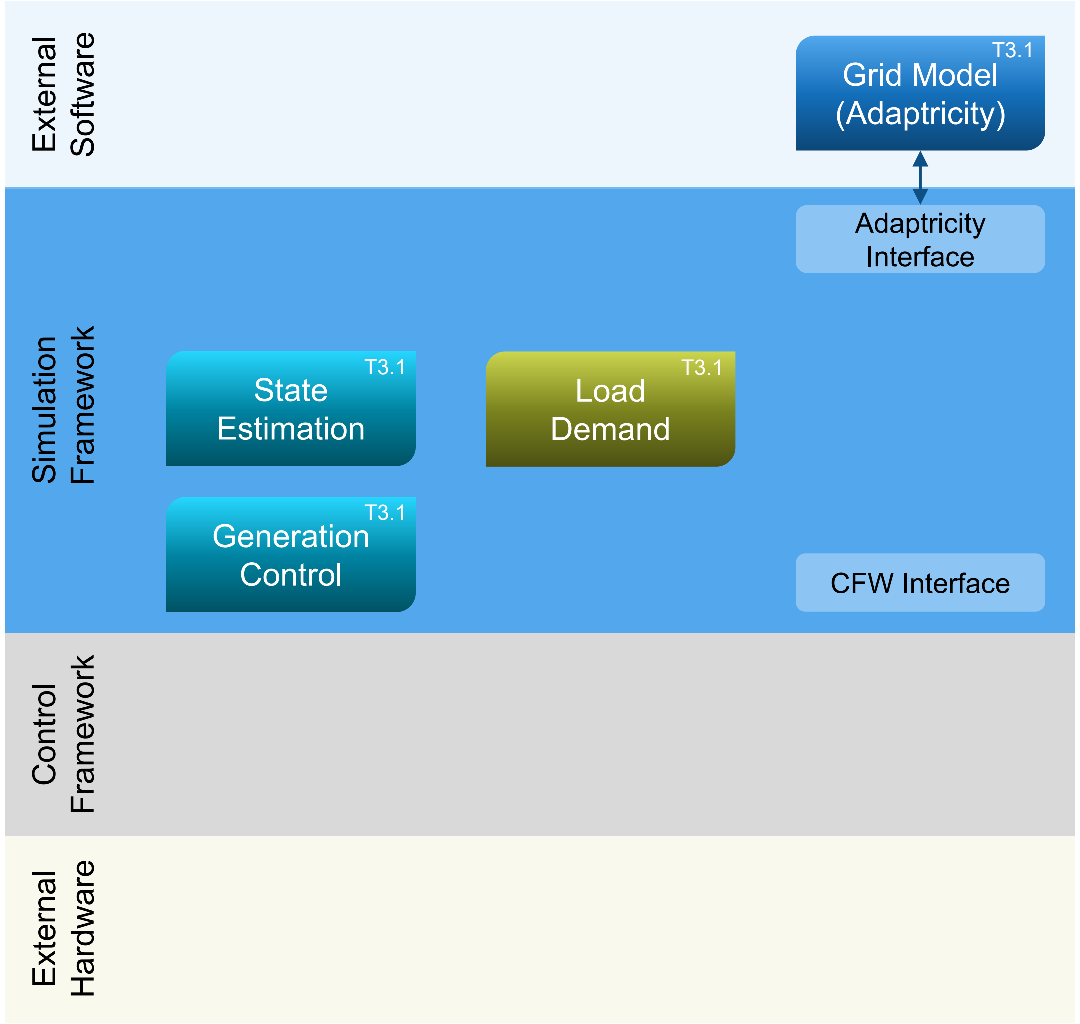
Update storage model without rewriting

```
    def run_storage_model(self): # Include losses into predefined storage model
        self.account_for_losses()
        super().run_storage_model()
```

Agenda

1. What is the ReMaP Simulation Framework (SFW)
2. Features of the SFW
3. Basic software design of the SFW
4. **Reference to already implemented “experiments” (use cases)**

ReMaP Task 3.1



Why

- Testing the online feedback optimization approach for real-time grid operation on a realistic simulation framework

What

- Simulation only (future connection to grid @Empa)
- Utilizing sparse real-time grid measurements to dynamically estimate the grid state, and choose the controllable generation set-points minimizing the operation cost while satisfying the grid constraints
- Grid modelled on Adaptiricty, power flow calculation via Adaptricity API

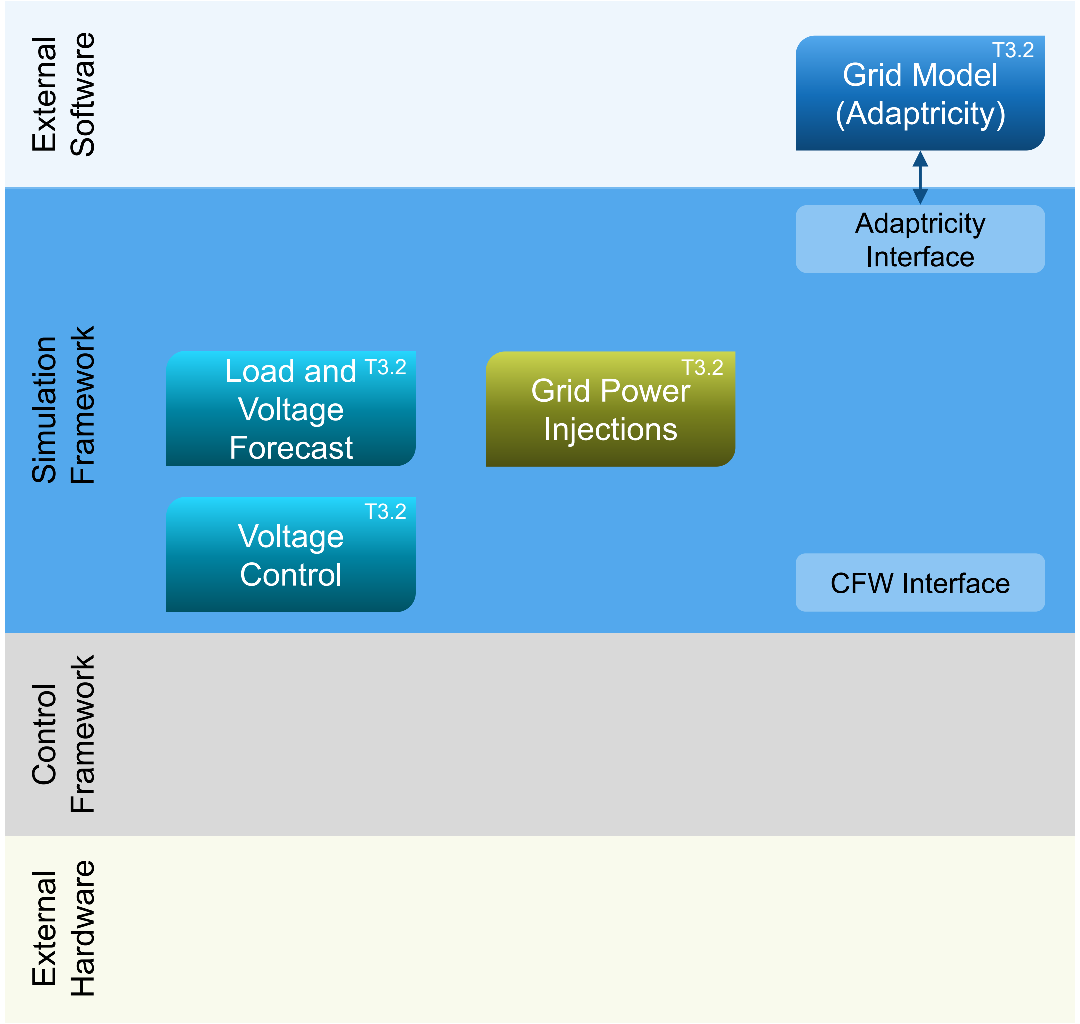
Who

- ETH Zürich, Automatic Control Laboratory

When

- August 2021 – ongoing

ReMaP Task 3.2



Why

– Developing a preventive control scheme based on thermostatically controlled loads and probabilistic predictions

What

– Simulation only
– Live data integration to follow
– Probabilistic load and voltage predictions for demand response purposes
– Grid modelled on Adaptiricty, timeseries power flow calculation via Adaptricity API

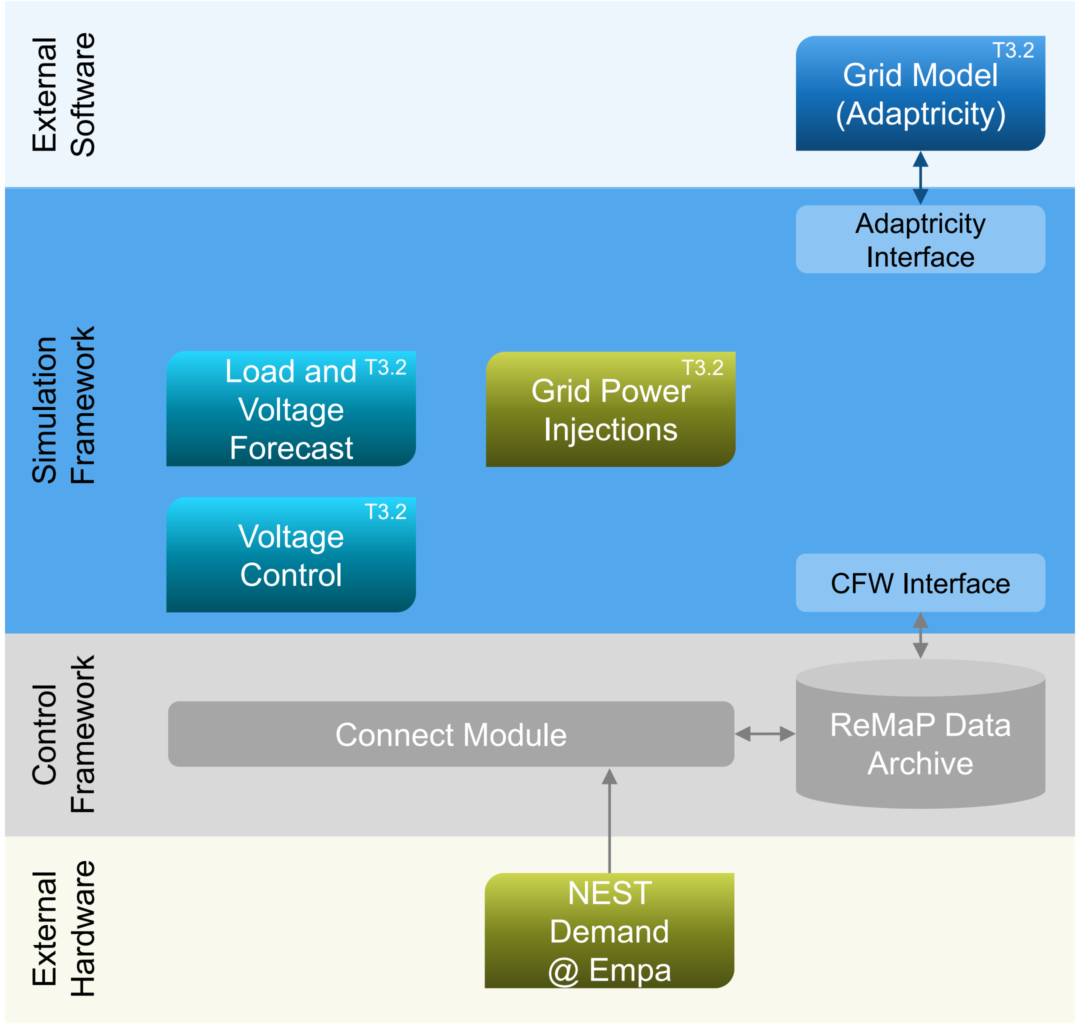
Who

– ETH Zürich, Power Systems Laboratory PSL

When

– August 2021 – ongoing

ReMaP Task 3.2 (Outlook)



Why

- Developing a preventive control scheme based on thermostatically controlled loads and probabilistic predictions

What

- Integrated demand data from NEST @Empa
- Probabilistic load and voltage predictions for demand response purposes
- Grid modelled on Adaptiricty, timeseries power flow calculation via Adaptricity API

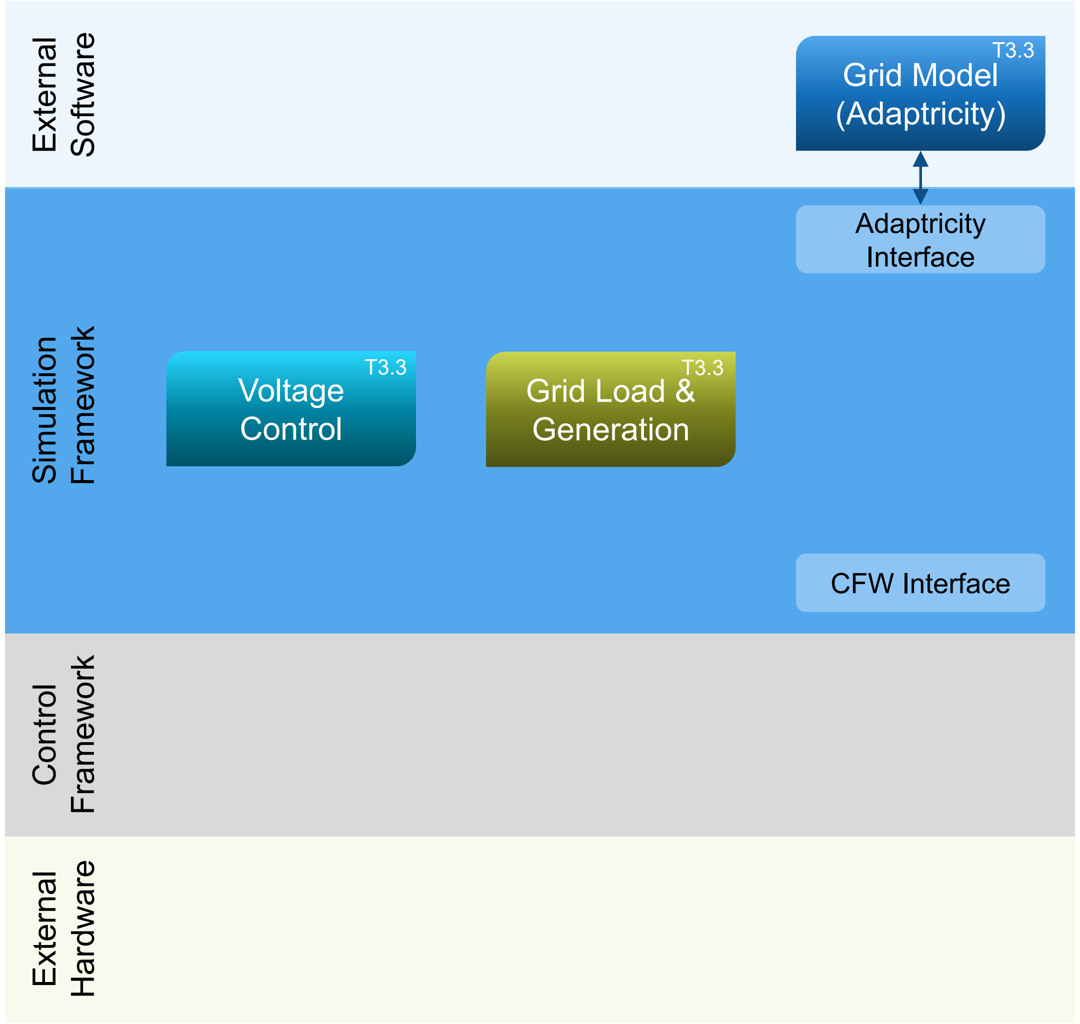
Who

- ETH Zürich, Power Systems Laboratory PSL

When

- August 2021 – ongoing

ReMaP Task 3.3



Why

- Quantify the impacts and benefits of active distribution networks on their hosting networks, considering the electrification of heating and mobility

What

- Simulation only
- Active distribution network control
- Controller changes the power output of controllable distributed energy resources whenever voltage or flow issues arise
- Grid modelled on Adaptiricty, power flow calculation via Adaptricity API

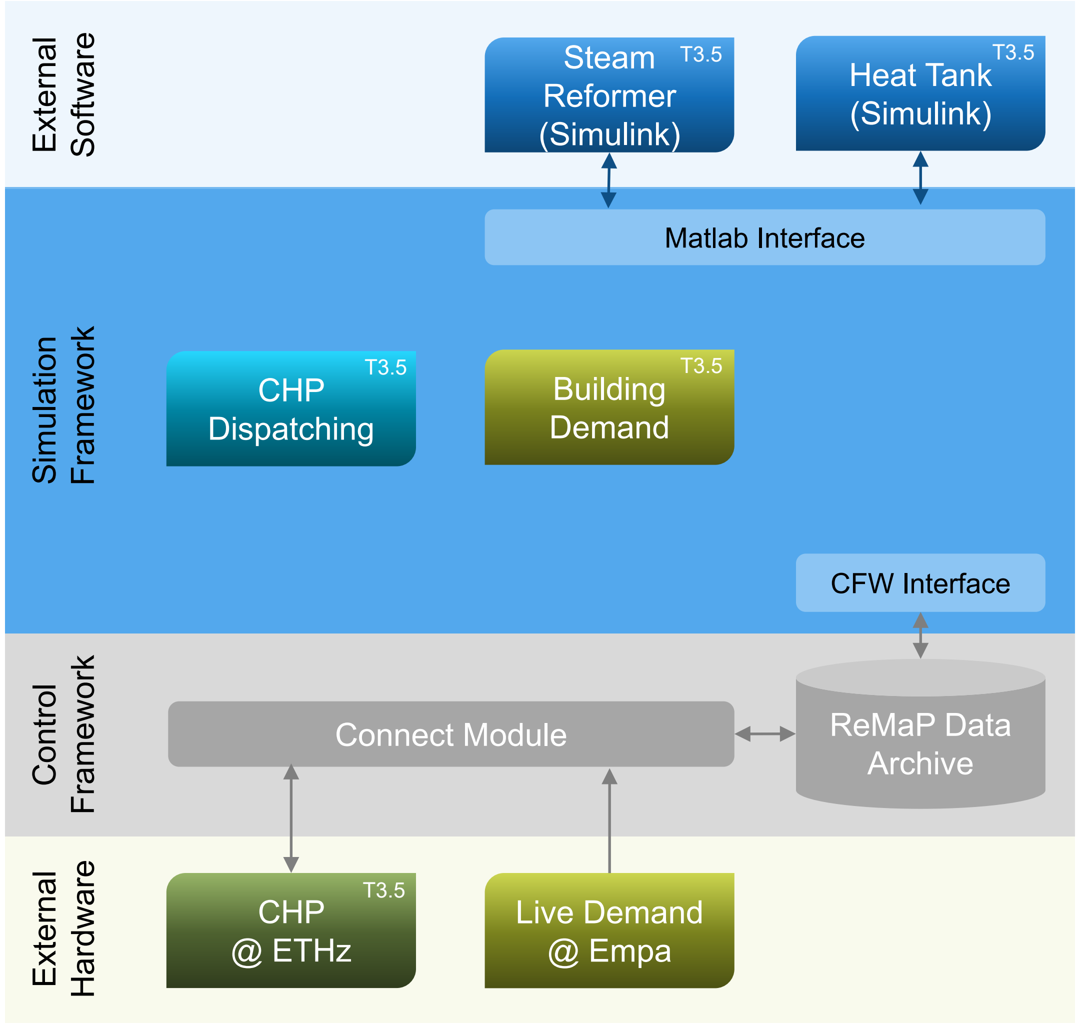
Who

- ETH Zürich, Reliability and Risk Engineering RRE

When

- July 2021

ReMaP Task 3.5



Why

- Evaluating system performance benefits from improved dispatching control and added components before modifying hardware

What

- Hardware-in-the-loop: CHP prototype at ETHz
- CHP plant, providing heat input for thermal storage and steam reformer models in Simulink
- HW dispatching through control algorithm in SFW based on model feedback and exogenous demand data

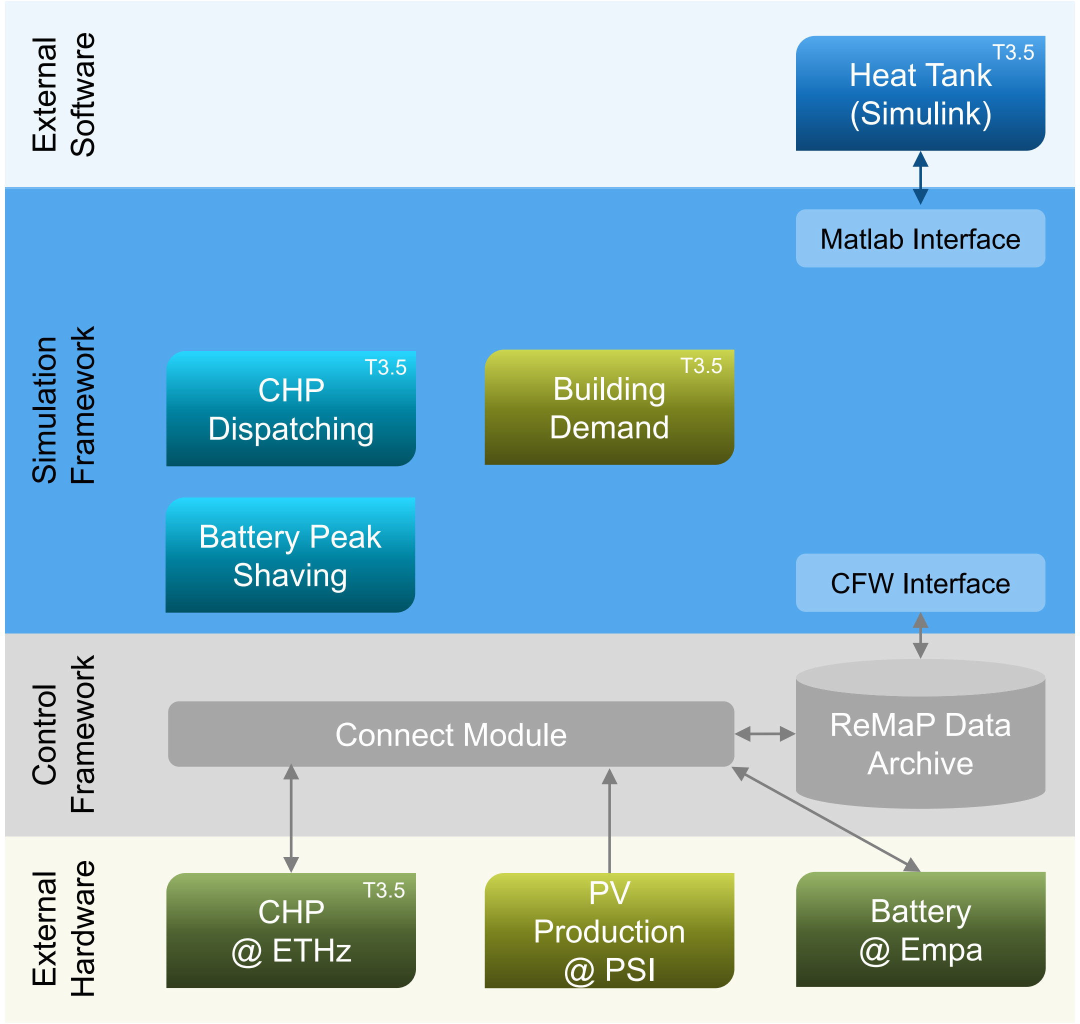
Who

- ETH Zürich, Laboratory of Aerothermochemistry and Combustion Systems LAV

When

- August 2020 – January 2021

ReMaP Task 3.8 SFW Showcase



Why

– Showcasing the functionality of the Simulation Framework and Control Framework

What

– Hardware-in-the-loop
– CHP plant, providing heat input for thermal storage to supply building heat demand
– Battery balances PV and CHP production and building electricity demand

Who

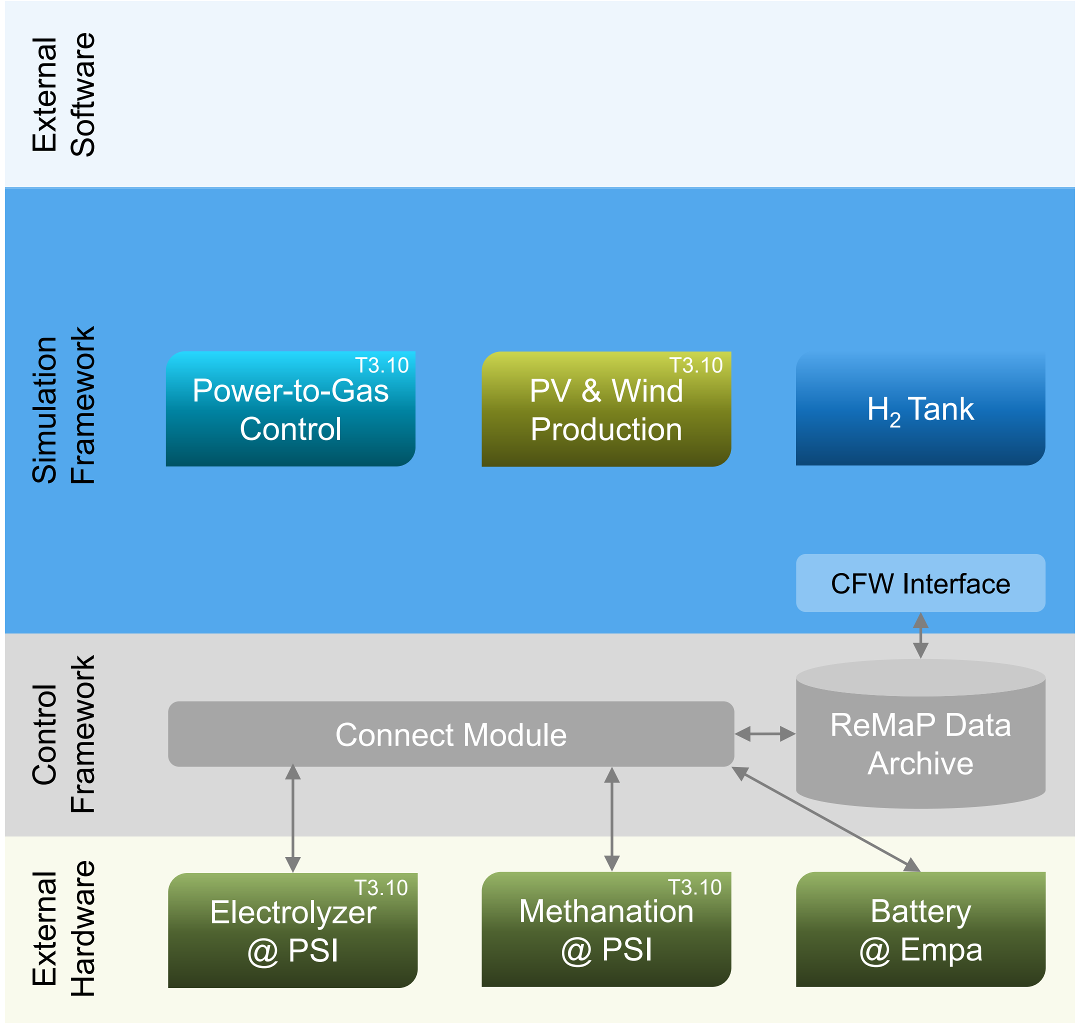
– ETH Zürich, Research Center for Energy Networks FEN
– Empa, Urban Energy Systems Laboratory

When

– November 2020 – December 2020

Data Controller Model Hardware

ReMaP Task 3.10, Case 1



Why

- H2 Mobility without using winter-electricity (seasonal storage)

What

- Hardware-in-the-loop: electrolyzer at PSI (ESI platform), battery at Empa, methanation reactor at remote location
- H_2 tank model virtually tracking hydrogen balance
- HW dispatching and control through control algorithm in SFW based on model feedback and exogenous demand data

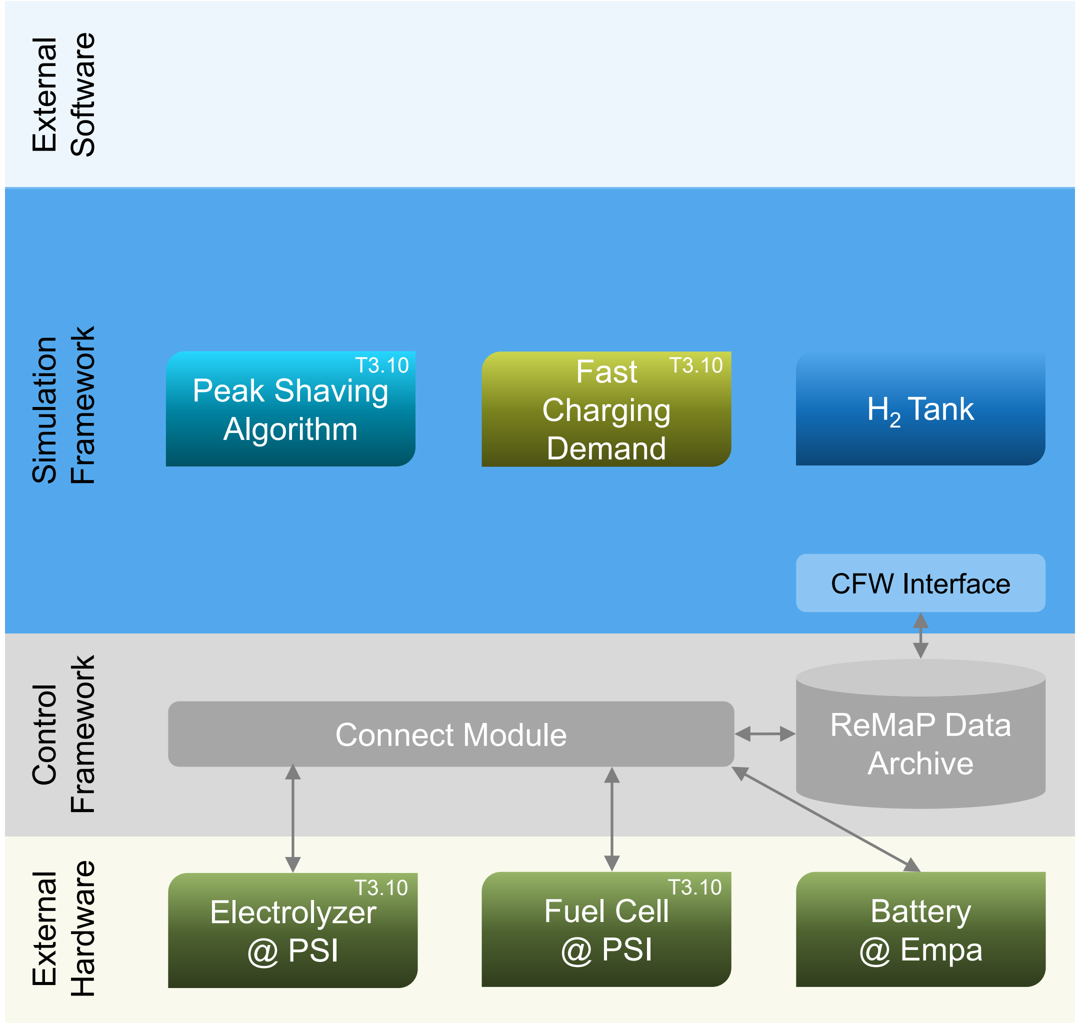
Who

- PSI, Bioenergy and Catalysis Laboratory LBK
- PSI, Electrochemistry Laboratory LEC

When

- June 2021 – July 2021

ReMaP Task 3.10, Case 2



Why

- Investigating peak shaving algorithms for electric vehicle fast charging stations

What

- Fast charging station peak shaving with power-to-gas-to-power
- Hardware-in-the-loop: electrolyzer and fuel cell at PSI (ESI platform), battery at Empa
- H_2 tank model virtually tracking hydrogen balance
- HW dispatching and control through control algorithm in SFW based on model feedback and exogenous demand data

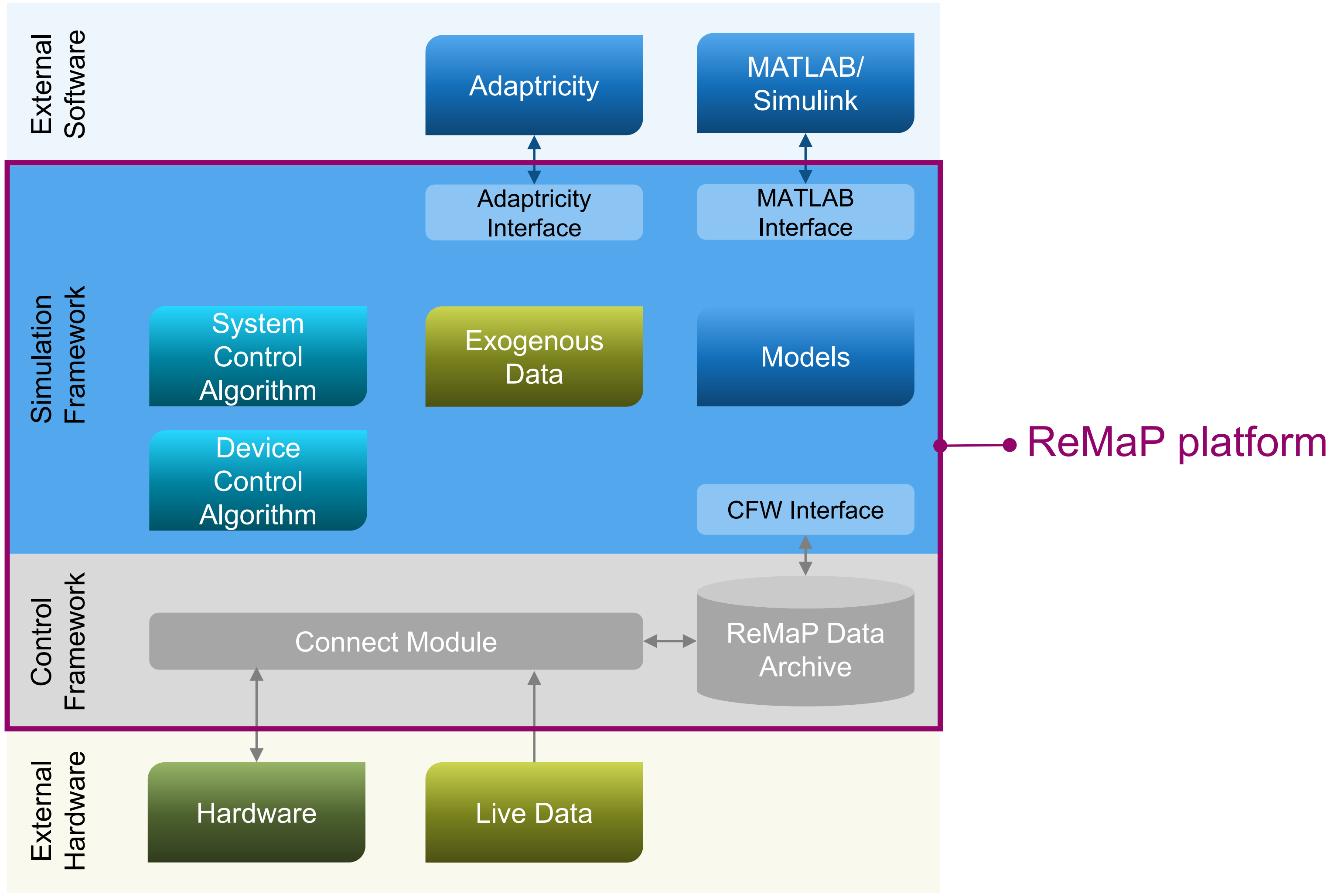
Who

- PSI, Electrochemistry Laboratory LEC

When

- June 2021 – September 2021

Simulation Framework as part of the ReMaP platform



Data Controller Model Hardware

ETH zürich

Dr. Adamantios Marinakis, Principal Expert, marinakis@fen.ethz.ch
Mr. Philippe Buchecker, bucheckp@fen.ethz.ch

ETH Zurich
Research Center for Energy Networks (FEN)
SOI C7
Sonneggstrasse 28
8092 Zürich, Switzerland

www.fen.ethz.ch