INL/MIS-22-66314

IES
Integrated Energy Systems

FORCE – Transient Physical Modeling Workshop

Hybrid for Analysts

Prepared and Presented by:

Dr. Daniel Mikkelson and Dr. Konor Frick

Session Agenda

1. Model development by integration of existing models (20 min)
   a) Drag-and-drop
   b) Passing parameters
2. Repeatably configurable modeling (20 min)
   a) Dymola parameter sweep
   b) Importing new initial conditions
   c) Manually changing initial conditions
   d) RAVEN interface
3. FMI/FMU in Dymola (20 min)
   a) Importing
   b) Exporting

IES Integrated Energy Systems

Integration of Existing Models

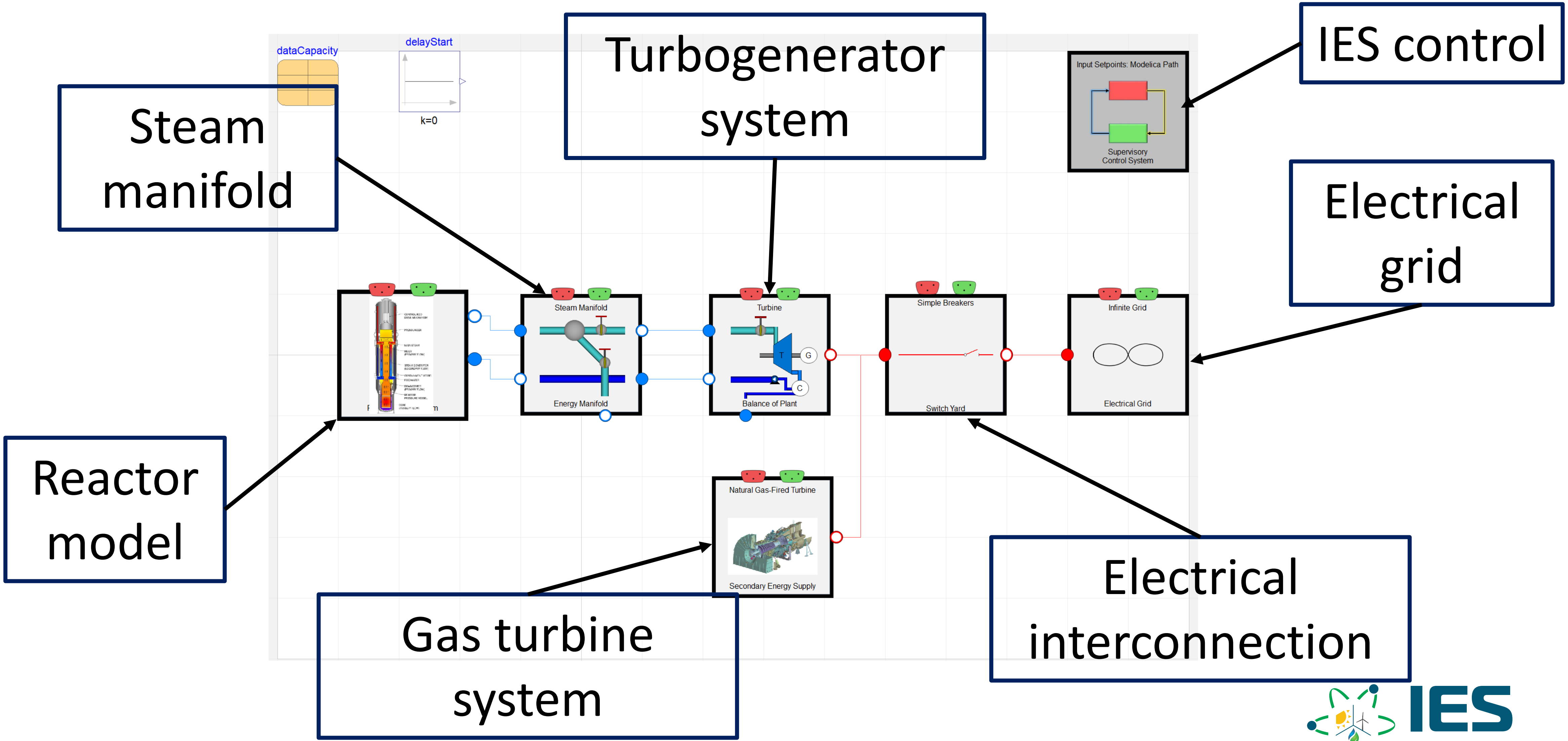
- Drag and drop of models is the most common method of building top-level systems
  - Example: IES, Reactor model
  - Prebuilt models combined in unique ways for simulation setup
  - Primary simulation difficulties are system-wide initialization and proper calibration of controls
- Subcomponents can be combined to make usable components
  - Example: Shell and tube heat exchanger
  - Configured models allow for standardized components for full system builds
  - Primary difficulty is to ensure appropriate parameter pass-through

Integration of Existing Models

- Using existing models takes advantage of object building within Modelica
- The same components can be used repeatedly
- Subsystems have been tested and verified
- Ports impose consistent communication between components

IES
Integrated Energy Systems

Example: IES

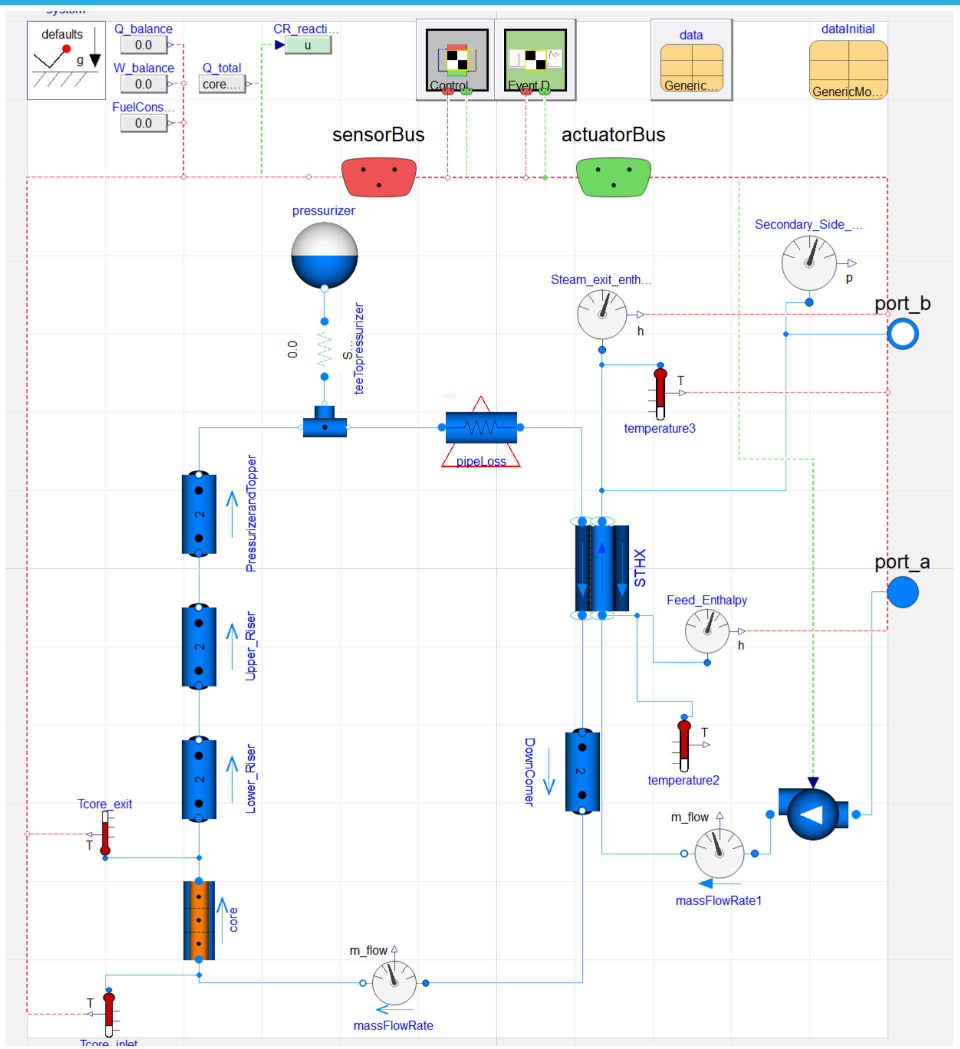


IES Integrated Energy Systems

Example: IES – Reactor Model

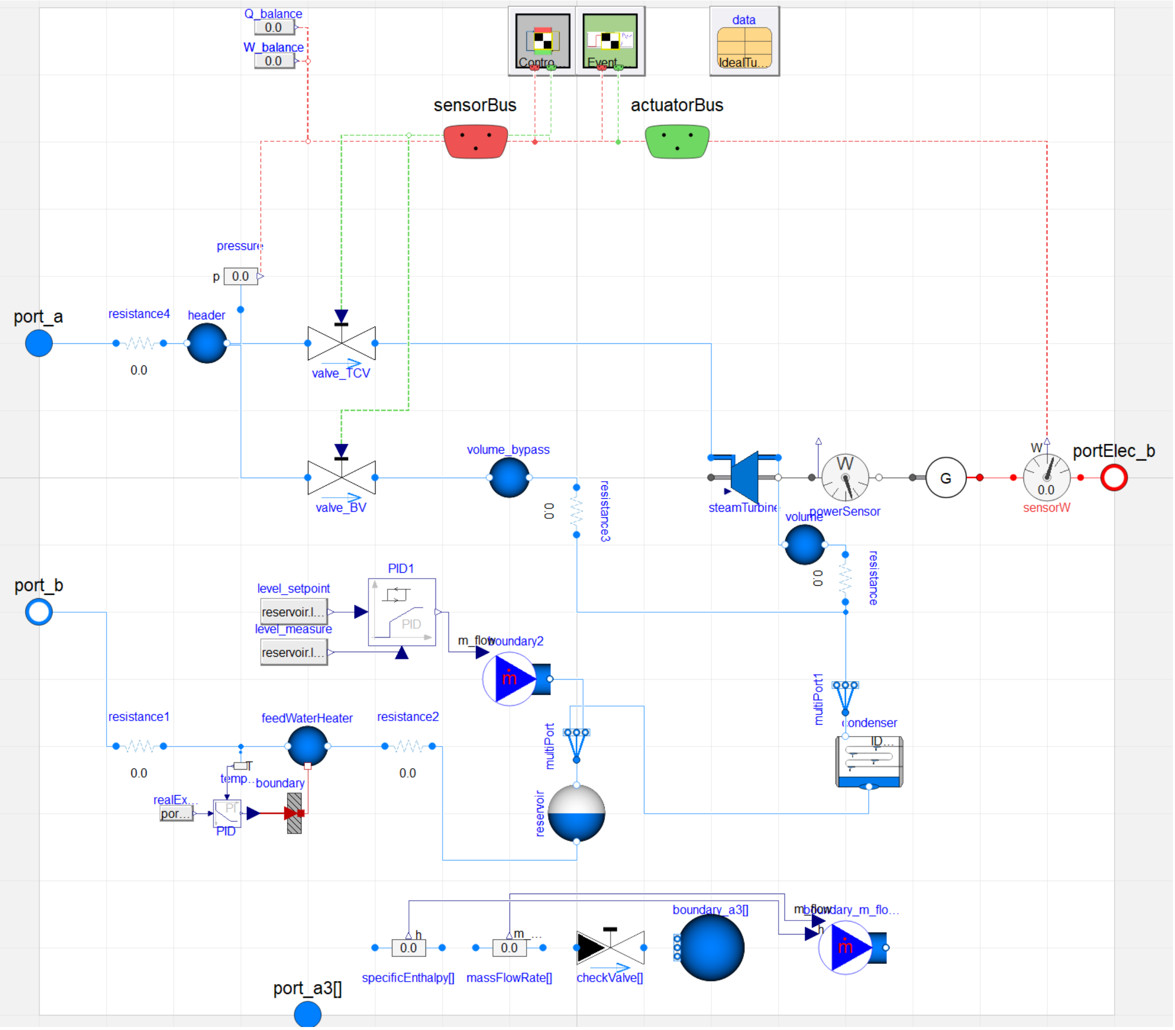
- Sixteen different drag-and-drop components make up reactor model
  - Includes pipes, sensors, feedwater pump, primary heat exchanger, nuclear core model, and control signals
  - Some of these models have drag-and-drop subcomponents

- Subsystem level is self-contained, only needing feed flow and steam produced connections.



Example: IES – Turbogenerator

- Turbogenerator system demonstrates five connection types
  - Fluid (blue)
  - Heat (red solid)
  - Mechanical (gray)
  - Electrical (red solid)
  - Control (red & green dashed)



7

Integration of Existing Models

- Construction using pre-existing models creates instantiations of the objects within current level model

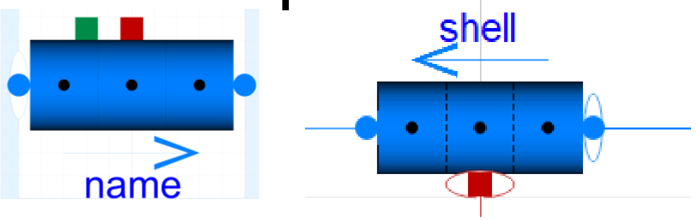
- Typically, ports and connectors are used to communicate information between objects

- Assuming the building block models exist, the construction process can happen quite intuitively
  - Example: Shell & tube heat exchanger

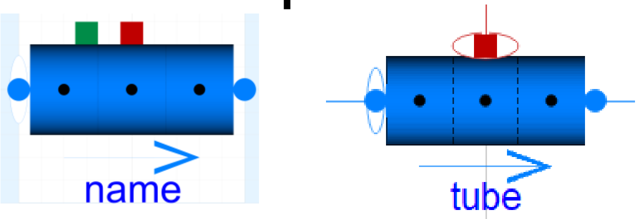
- Parameter passing must be handled at every level

Example: Shell & Tube Heat Exchanger

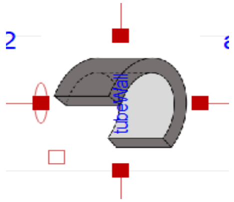
- What do we need to make a STHX?
  - Shell fluid flow path



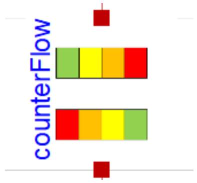
  - Tube fluid flow path



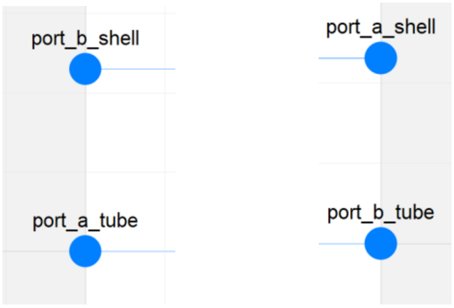
  - Pipe model to establish conductivity



- Possibly a vectorization reversing unit to allow for counter-flow OR concurrent flow



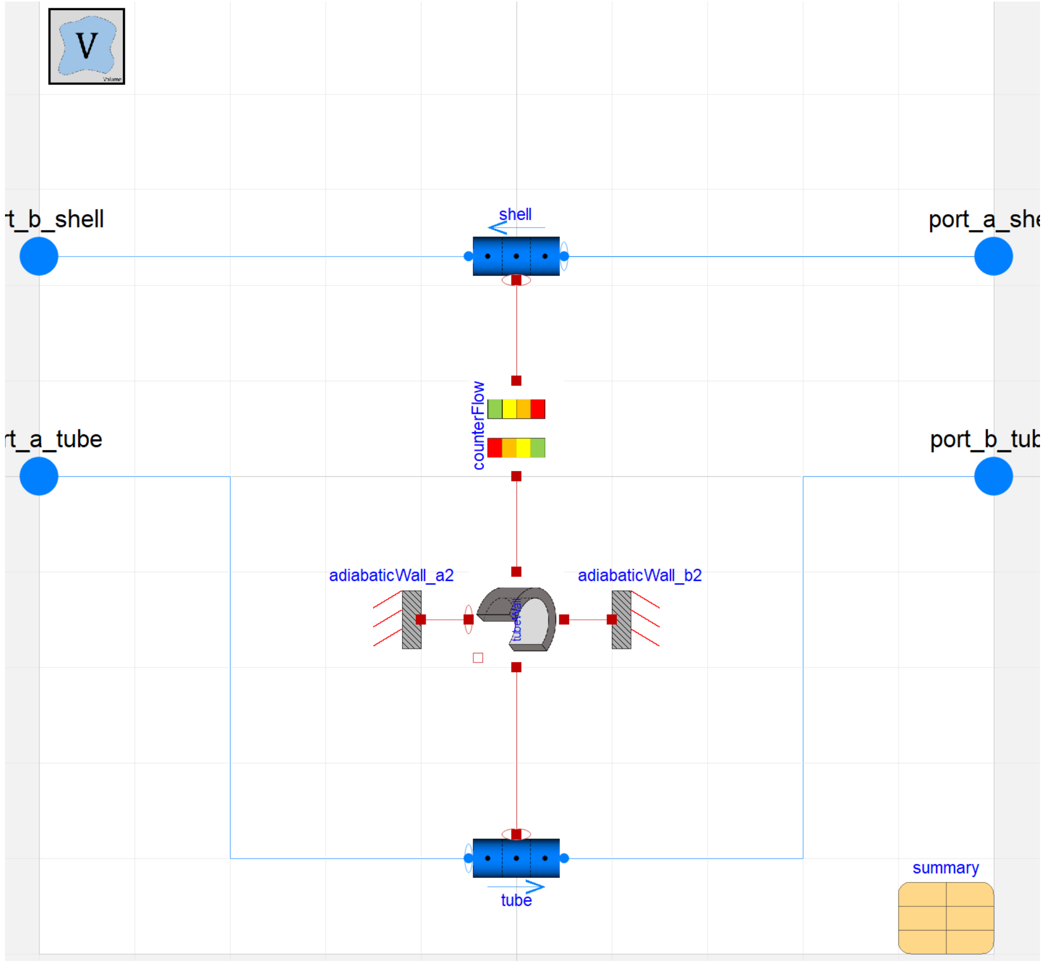
- External fluid connectors



Example: Shell & Tube Heat Exchanger

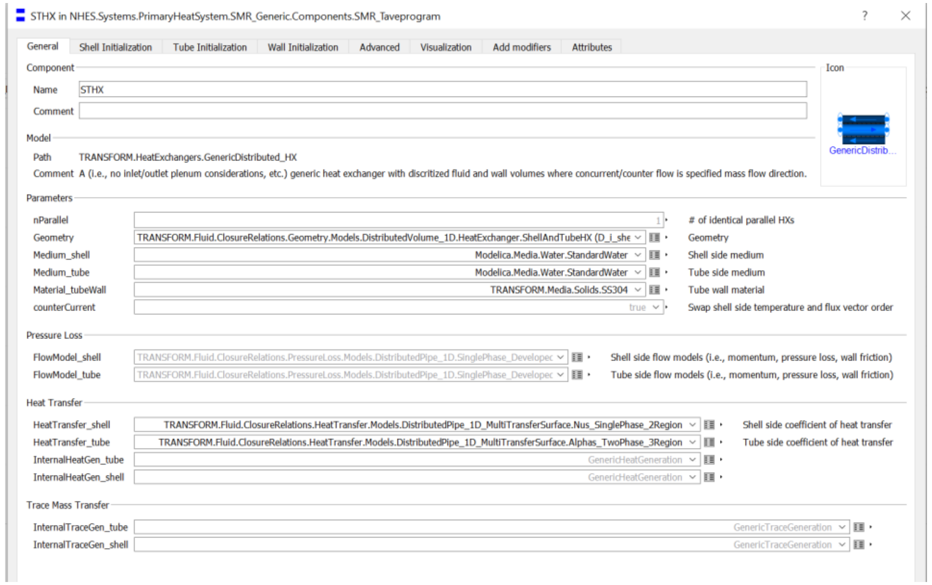
- Finished product thermally connects two fluid streams

- One final question: how do we properly pass parameters to next-level modeling?
  - Each component in the figure on the right has its own parameters
  - For example: what is the diameter of the tube in the tube model?



Integration of Existing Models: Passing Parameters

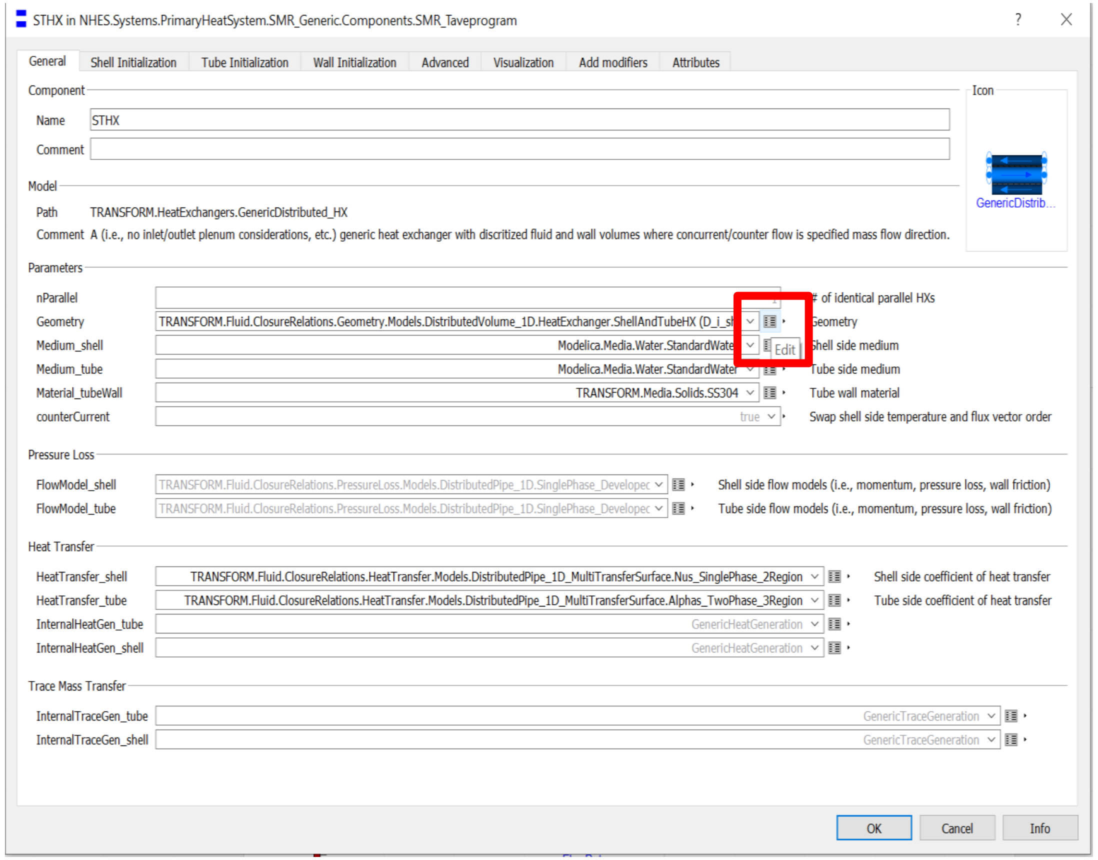
- Typically, parameters must be re-declared at every level
  - Default values can be put in, as the highest modeling level will be distributed down
- “Replaceable” keyword allows for all potential values matching the type of that parameter to be selected via drop-down menu
  - For example: two-phase media types
- Parameters can be grouped into data structures for easier pass-through

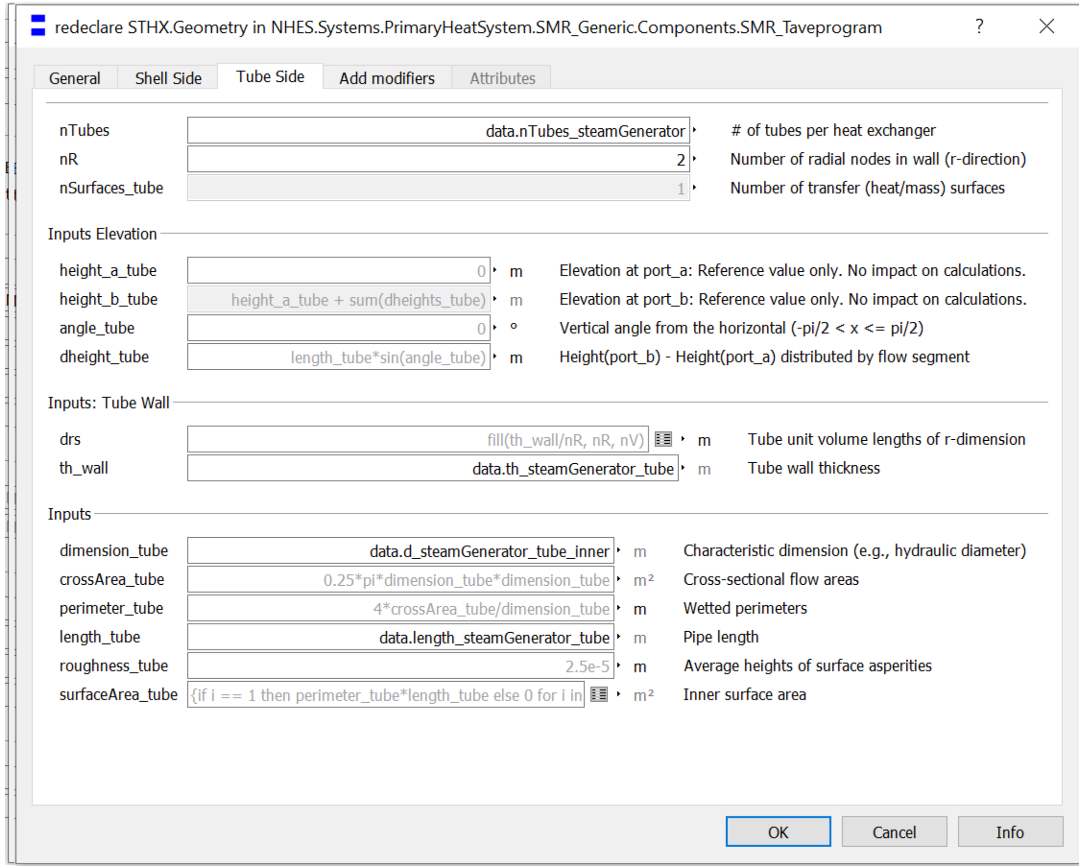


Parameter interface seen above. Interface method depends on type of parameter (single value, package selection, set of values, etc)

IES
Integrated Energy Systems

Passing Parameters





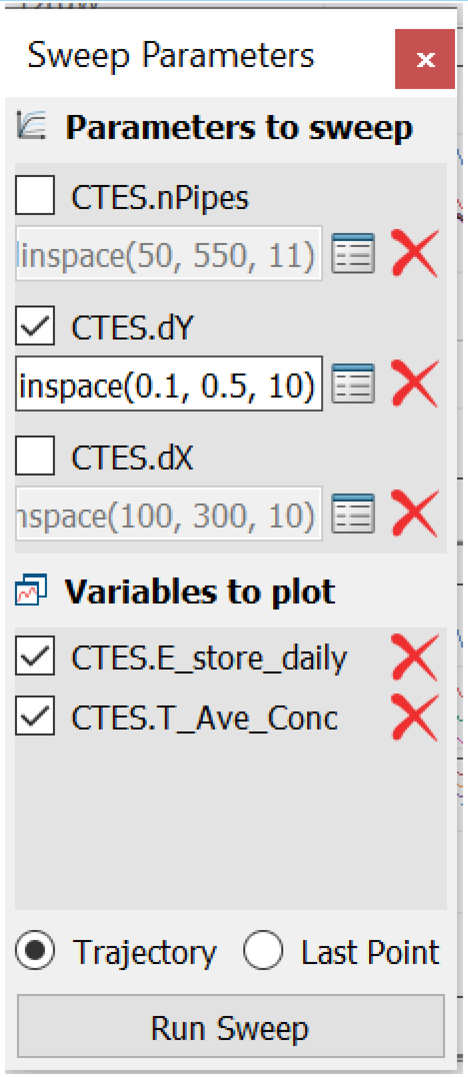
Integration of Existing Models

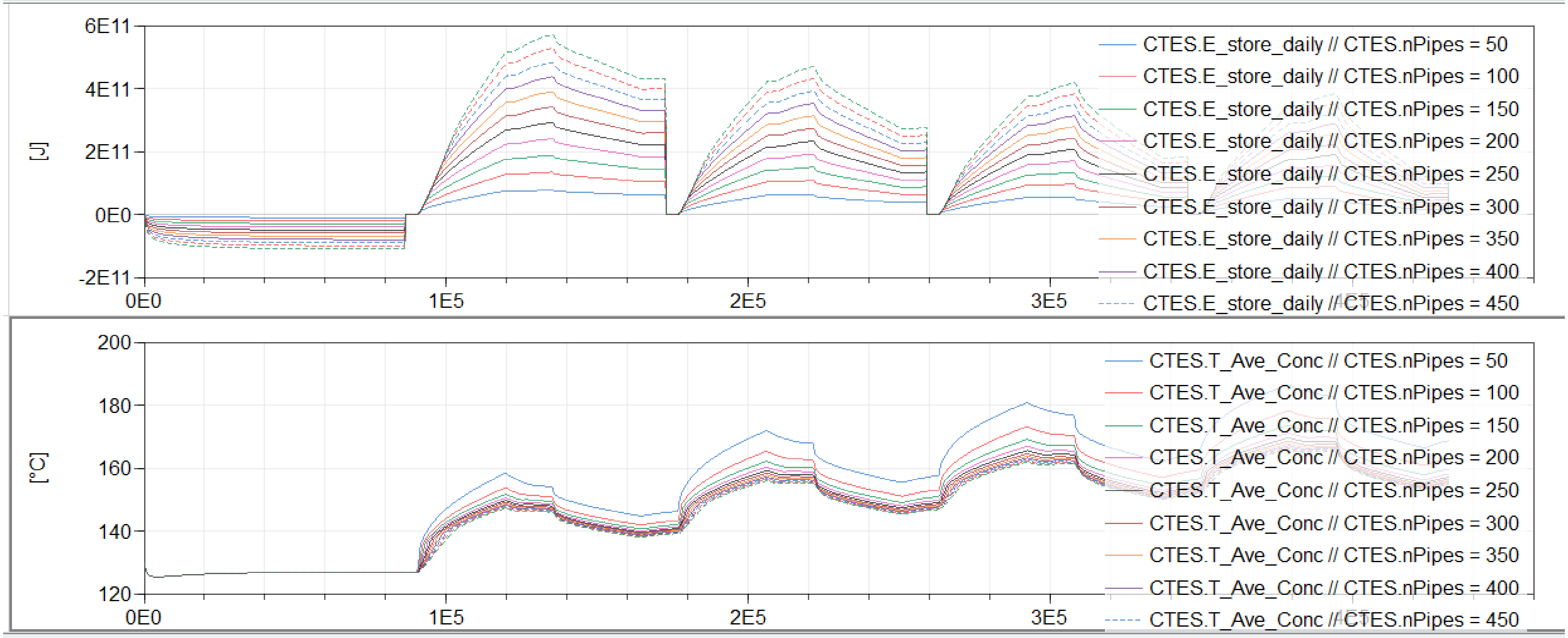
- Using existing models takes advantage of object building within Modelica
- The same components can be used repeatedly
- Ports impose consistent communication between components
- When building sub-models and subsystems, make sure that relevant parameter passing methods are set up

IES
Integrated Energy Systems

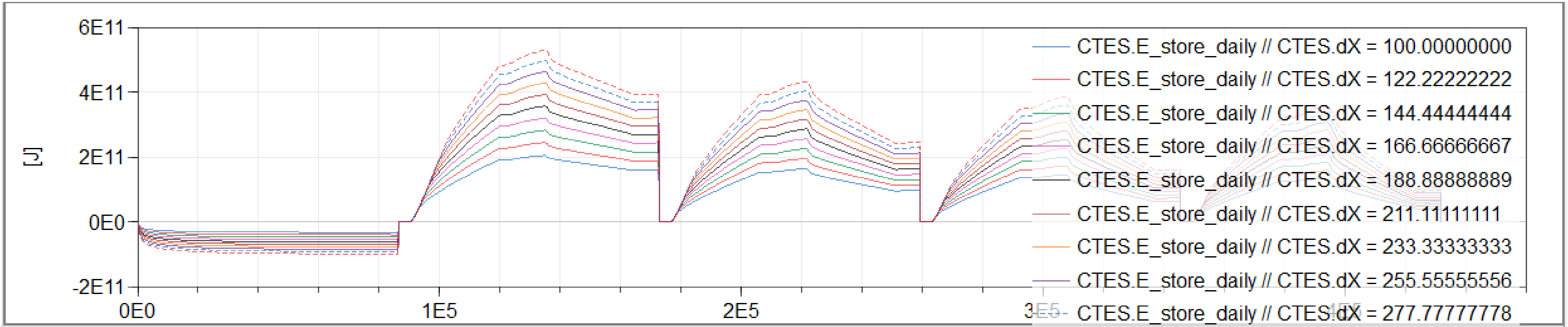
Parameter Sweeping

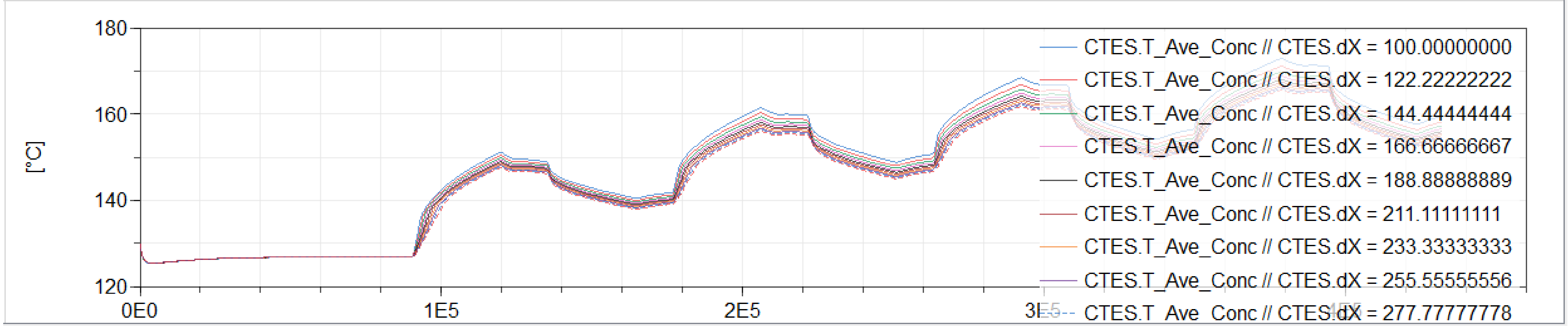
- Dymola has internal parameter sweeping method
- Allows for output space generation across single altered parameter at a time
- Auto generates separate output files to keep post simulation
- Auto generates plotting set of output values desired by user



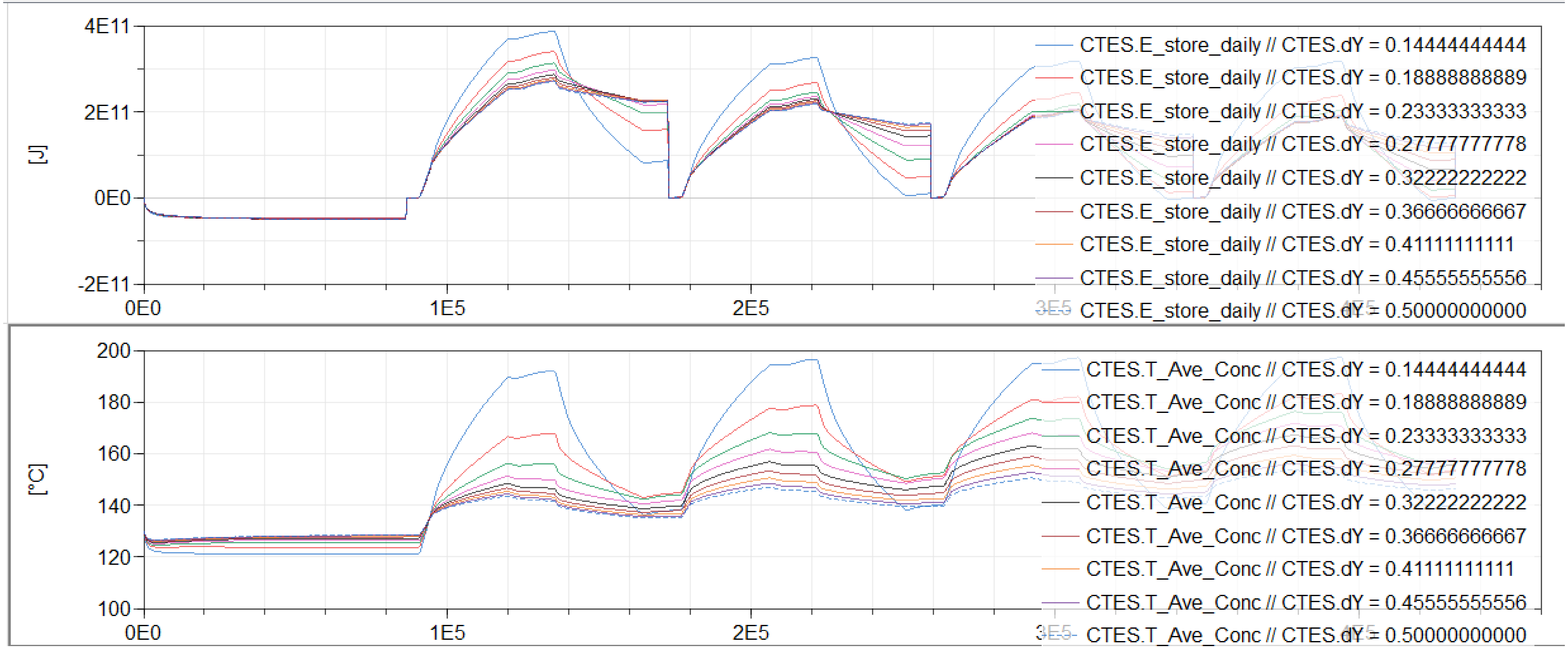


IES
Integrated Energy Systems





IES
Integrated Energy Systems



IES
Integrated Energy Systems

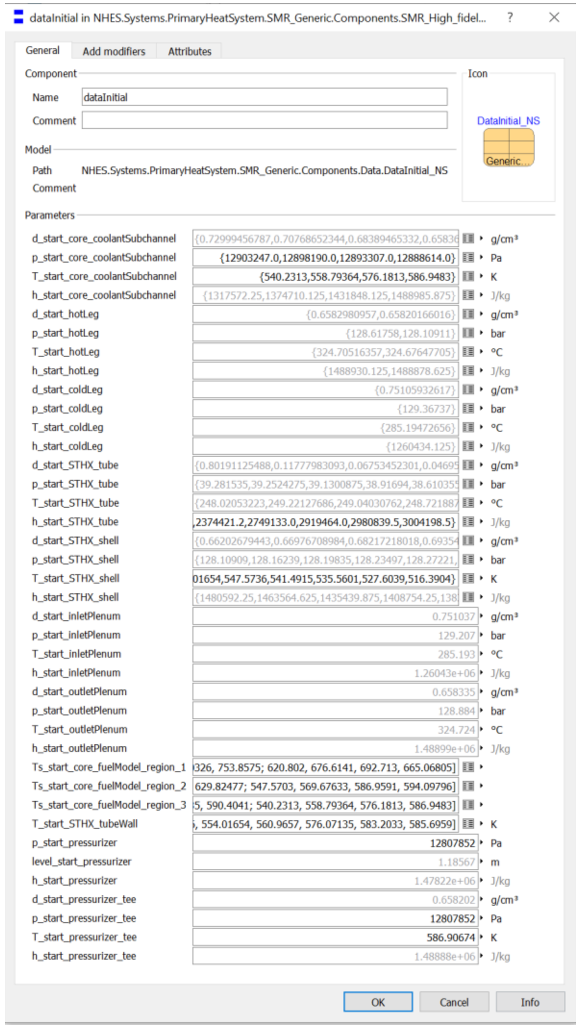
Manual Parameter Sweep

- In the case that a model is sensitive to initial conditions, it is possible to manually alter parameters via the dsfinal.txt file to effectively manually parameter sweep

- Combined with script generation, this process can be automated so that there is less user attention required

Initial Conditions

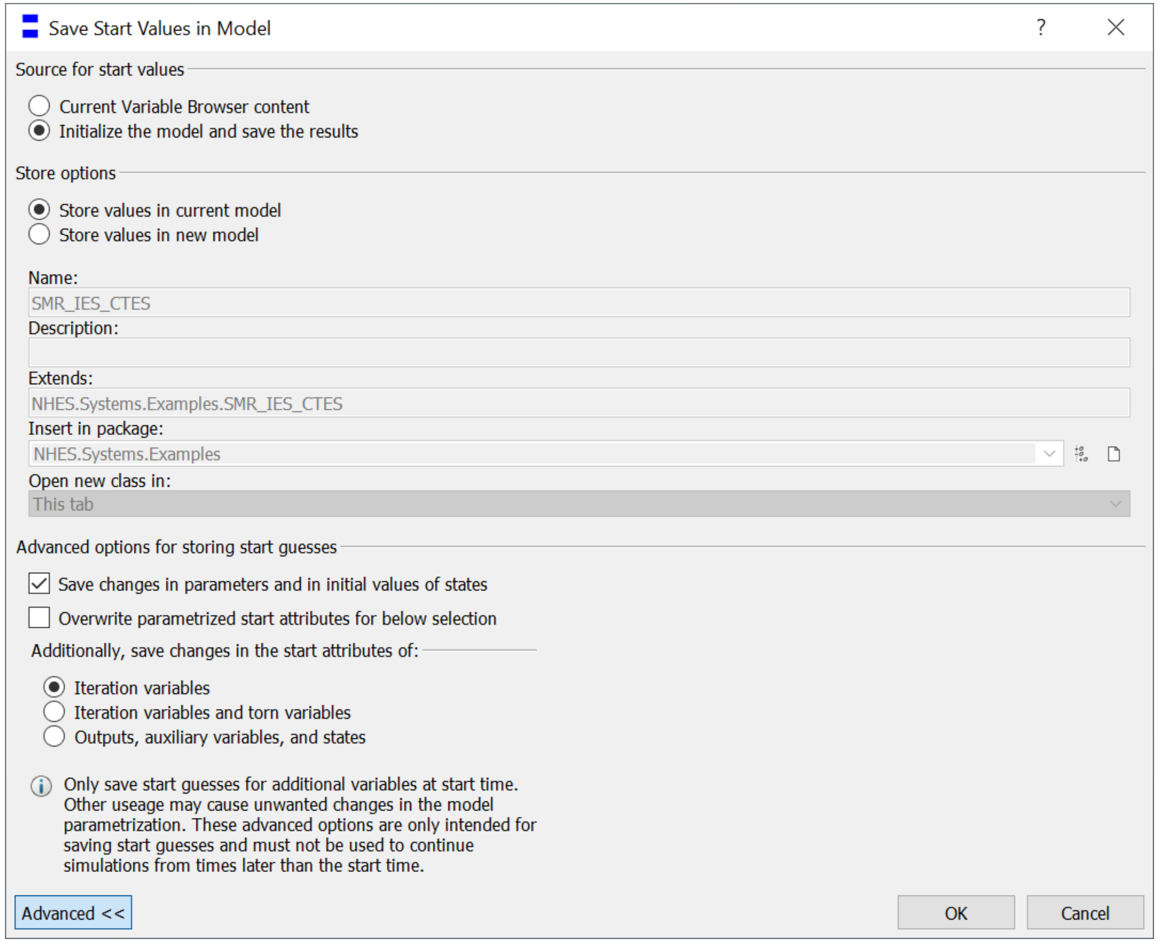
- Model initialization is key to obtaining results, especially if a simulation stabilization time frame can be avoided
- One method of creating an initial state is to use a robust outside result to create an initial conditions table
    - ASPEN HYSYS is often used



IES Integrated Energy Systems

Initial Conditions

- Inherent method within Dymola to save within a model the initial conditions
- When used, the output space is saved within the model directly as adjustments to the attributes



Initial Conditions

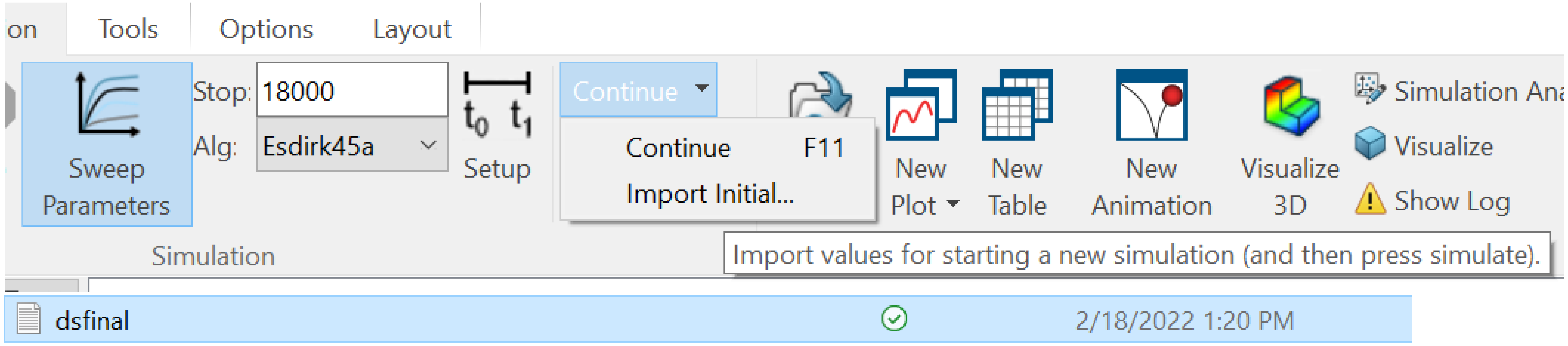
```
BOP(
  CS(
    BV_openingNominal(k(start=0.001)),
    PID_BV_opening(
      I(k(start=0.1)),
      addP(
        k1(start=1.0),
        u1(start=0.0300000001485),
        u2(start=0.0300000001485)),
      gainPID(k(start=-1.0)),
      gainTrack(k(start=-1.1111111111111112)),
      gain_u_m(k(start=5E-10)),
      gain_u_s(k(start=5E-10)),
      limiter(uMax(start=0.999), uMin(start=-0.0009)),
      null_bias(k(start=0.0)),
      yMax(start=0.999),
      yMin(start=-0.0009)),
    PID_TCV_opening(
      I(k(start=2.0)),
      addP(
        k1(start=1.0),
        u1(start=0.0086185),
        u2(start=0.0086185)),
      gainTrack(k(start=1.1111111111111112)),
      gain_u_m(k(start=2.5E-09)),
      gain_u_s(k(start=2.5E-09)),
      limiter(uMax(start=0.5), uMin(start=-0.4999)),
      null_bias(k(start=0.0)),
      u_s(start=3447400.0),
      yMax(start=0.5),
      yMin(start=-0.4999)),
    TCV_openingNominal(k(start=0.5)),
    delayStartBV(start=100.0),
    p_Nominal1(k(start=3447400.0)),
    switch_P_setpoint(y(start=60000000.297)),
    valvedelay(k(start=100.0)),
    valvedelayBV(k(start=100.0))),
  PID(
    I(k(start=2.0)),
    addP(
      k1(start=1.0),
      u1(start=1.0),
      u2(start=1.0)),
    gainPID(k(start=100000000.0)),
    gainTrack(k(start=1.1111111111111112E-08)),
    gain_u_m(k(start=0.002374343174368348)),
    gain_u_s(k(start=0.002374343174368348)),
    k_m(start=0.002374343174368348),
    k_s(start=0.002374343174368348),
    limiter(
      u(start=100000000.0),
      uMax(start=1E+60),
      uMin(start=-1E+60)),
    null_bias(k(start=100000000.0)),
    u_m(start=421.1691093331664),
    yMin(start=-1E+60)),
  PID1(
    PID(
      I(k(start=2.0)),
      Nd(start=10.0),
      Ni(start=0.9),
      Td(start=0.1),
      Ti(start=0.5),
      addP(
        k1(start=1.0),
        u1(start=1.0),
        u2(start=1.0)),
      gainPID(k(start=100.0)),
      gainTrack(k(start=0.011111111111111112)),
      gain_u_m(k(start=0.1))
```

```
actuatorBus(opening_BV(start=0.001), opening_TCV(start=0.5)),
boundary(port(T(start=421.1691093331664))),
boundary2(medium(
    T(start=298.16763607879363),
    T_degC(start=25.017636078793657),
    d(start=998.544943058541),
    p_bar(start=34.473800000000004),
    sat(Tsat(start=514.8425665422984)),
    u(start=104571.53362749857)), ports(h_outflow(start={108023.9370710939}),
```

```
 Kt(start=0.013324090093760938),
 Q_mech(start=65056859.70577499),
 Q_units(start={42261285.79911617,42261285.79911617}),
 Q_units_start(start={42261285.79911617,42261285.79911617}),
 Qbs(start={-9732855.946228676,-9732855.946228676}),
 T_a_start(start=293.15),
 T_b_start(start=293.15),
 T_nominal(start=293.15),
 bubble_in(d(start=820.3581983078773), h(start=1013666.6724914373)
 bubble_out(d(start=989.8436373961912), h(start=191812.29519356362
 d_nominal(start=13.671247252758716),
 dew_in(d(start=15.307197090608243), h(start=2803284.170249812)),
 dew_out(d(start=0.06816373081854721), h(start=2583886.8570257137)
 h_a_start(start=2997670.0),
 h_b_start(start=2058530.3155751962),
 h_is(start=2070197.5860370956),
 h_out(start=2209318.4481315315),
 p_a_start(start=3337380.0),
 p_b_start(start=10000.0),
 p_inlet_nominal(start=3337380.0),
 p_outlet_nominal(start=10000.0),
 p_ratio(start=0.0032668200354825697),
 portHP(
```

Initial Conditions

- Another method of importing initial conditions is using default output format



- This method can be used to alter parameters by altering the text file

IES Integrated Energy Systems

RAVEN Interfacing

- Default initialization or final status text file is standard RAVEN input method
- Values identified in RAVEN input substituted

```
-2  1.0000000000000001E-01       0                        0
 1   280 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.k
-2  5.0000000000000000E-01  9.9999999999999997E-61 1.0000000000000000E+100
 1   280 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.Ti
-2  1.0000000000000001E-01       0                 1.0000000000000000E+100
 1   280 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.Td
-2      0                        0                        0
 1   280 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.yb
-2  5.0000000000000000E-01       0                        0
 6   256 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.k_s
-2  5.0000000000000000E-01       0                        0
 6   256 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.k_m
-2  5.6600000000000001E+00       0                        0
 1   280 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.yMax
-2      0                        0                        0
 1   280 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.yMin
```

RAVEN Interface

- Executable made via Dymola and path input into RAVEN
  - User should make sure that “evaluate parameters at translation” option is disabled
- Dymola is subType “Dymola” in the input deck
- Input name type is “DymolaInitialisation”

IES Integrated Energy Systems

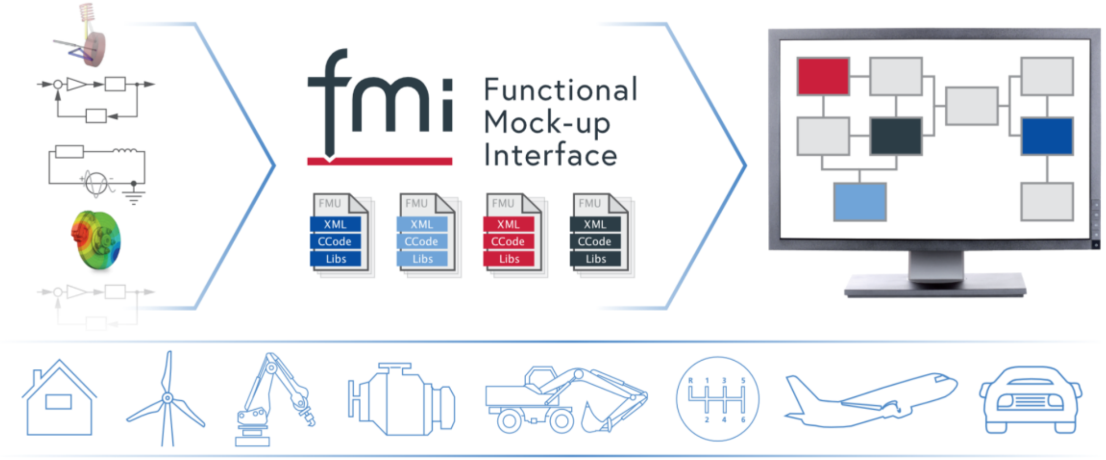
Model Analysis

- Scripting allows for manual creation of parameter sweep
  - Method is: Translate(), import(), simulate()
- Dymola has internal parameter sweep methods
  - Only one parameter can be changed at once
- RAVEN interface uses standard input to accept new parameter methods
- Reminder from previous: models can use text reading for input, which can read dispatch information generated by another code

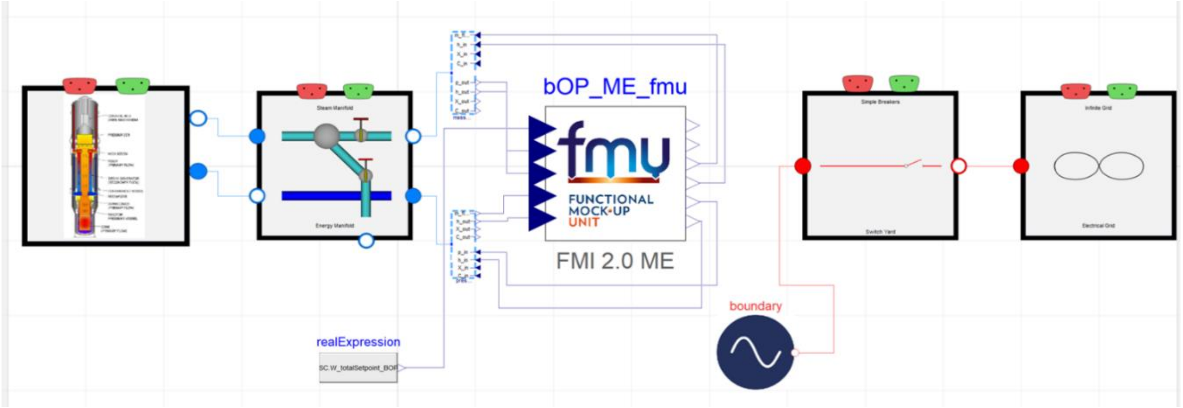
Functional Mockup Units

Functional Mockup Units

- Began in 2008 started by the MODELISAR project
- Standardized interface to be used in computer simulations to develop cyber-physical systems.
- Adopted by over 170+ toolsets with an actively maintained FMI/FMU import/export capability including:
  - ANSYS, Dymola, Matlab/Simulink, Java, Python, STARCCM+
- Capable of exporting models developed in another tool and simulating in an FMI/FMU compliant toolset without a customized API requirement.

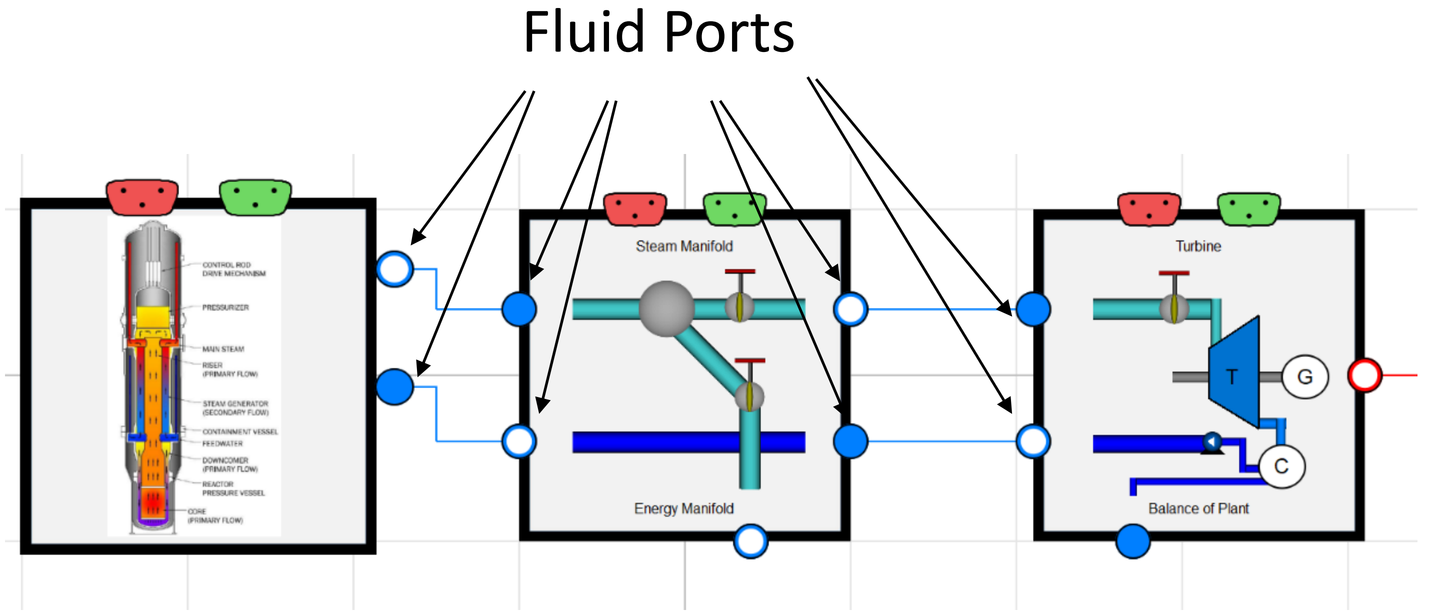


Picture from: https://fmi-standard.org/



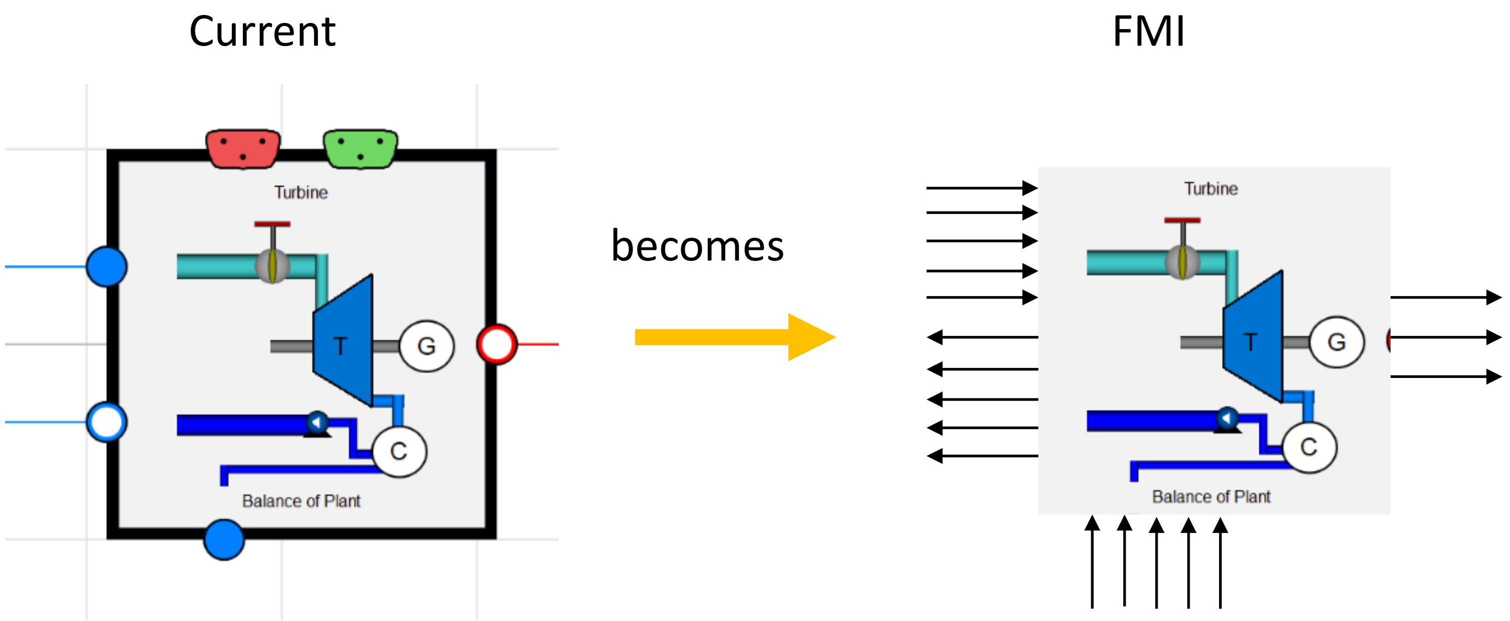
Current Dymola

- Each Fluid Port contains
  - Mass flow (Flow variable), m_flow
  - Conditional enthalpy (stream variable), h_outflow
  - Pressure, P
  - Trace Substance Fraction (stream variable), Ci
  - Mass Fraction (stream variable), Xi
- Stream variables are only used if m_flow < 0.



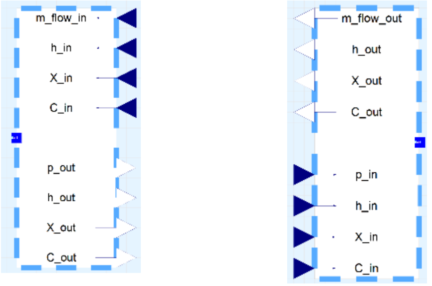
FMI/FMU Signals

- FMI/FMUs operate only with input and output signals. Therefore, each FMI/FMU model would need to be designed to modify “fluid and electric ports” into input and output signals.
- This required the creation of a FMI/FMU connector package within the Hybrid repository.

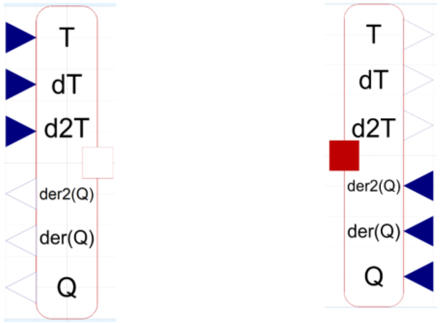


Standardized FMI/FMU Adaptors

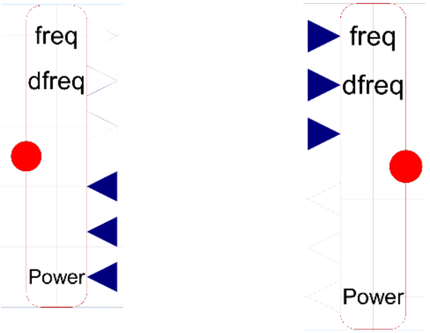
Fluid Adaptors

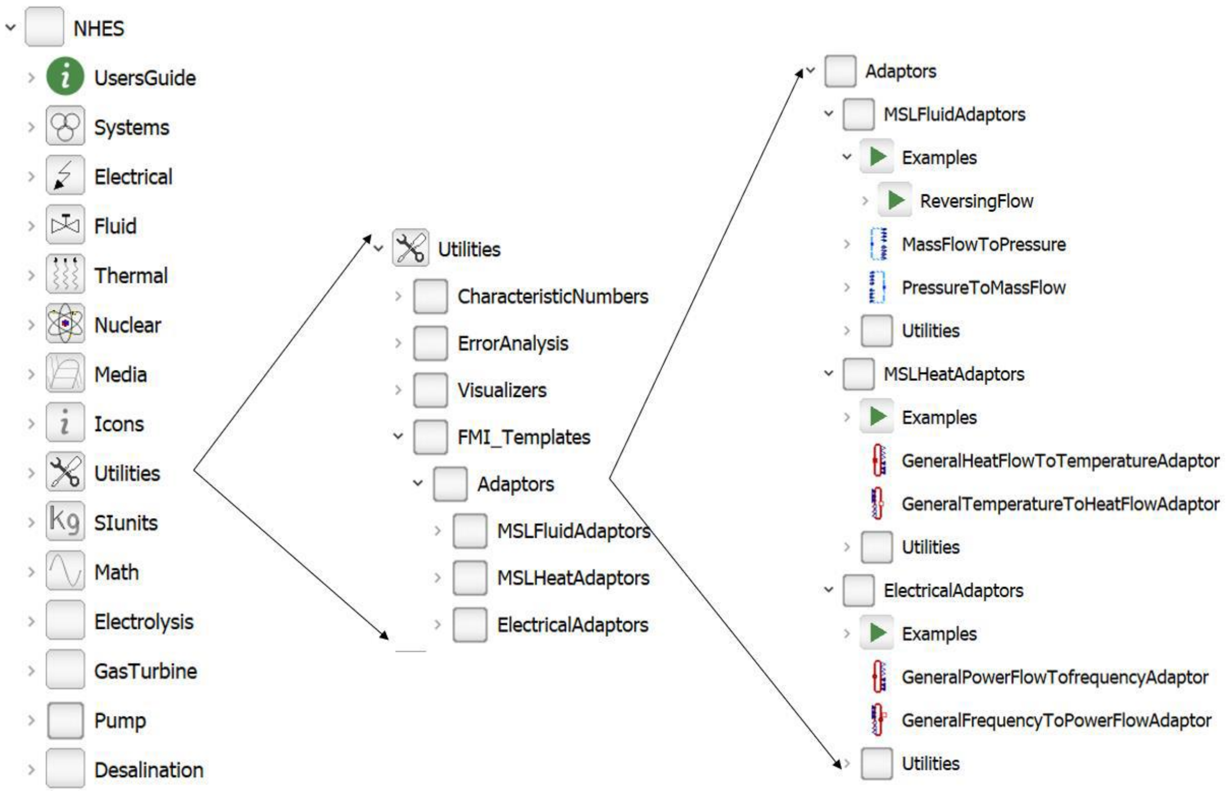


Heat Adaptors

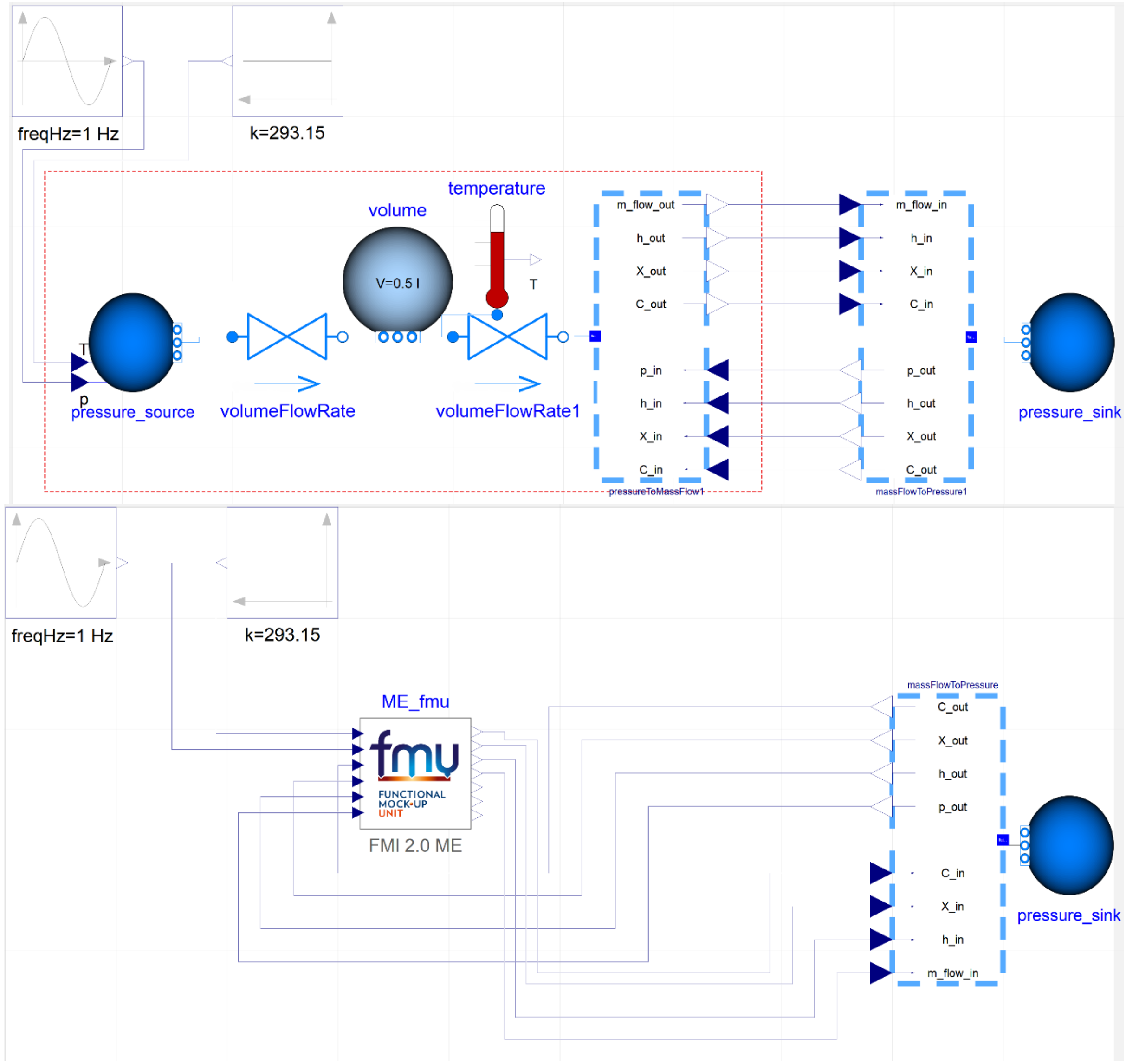


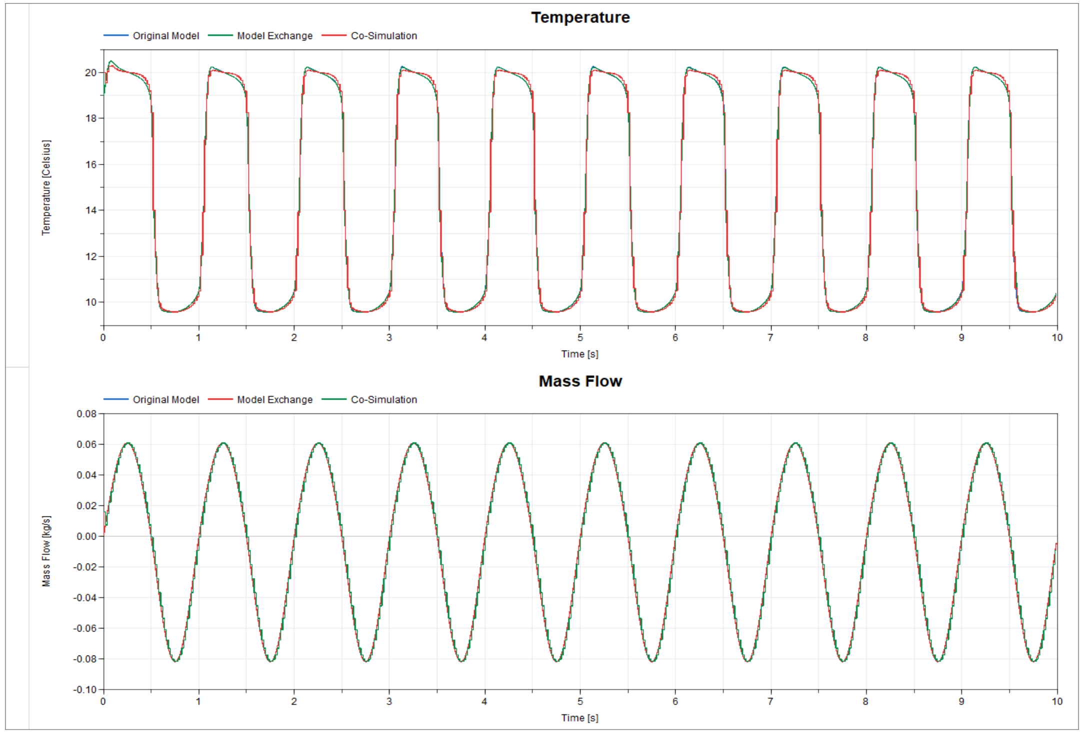
Electrical Adaptors





FMI/FMU Demonstration



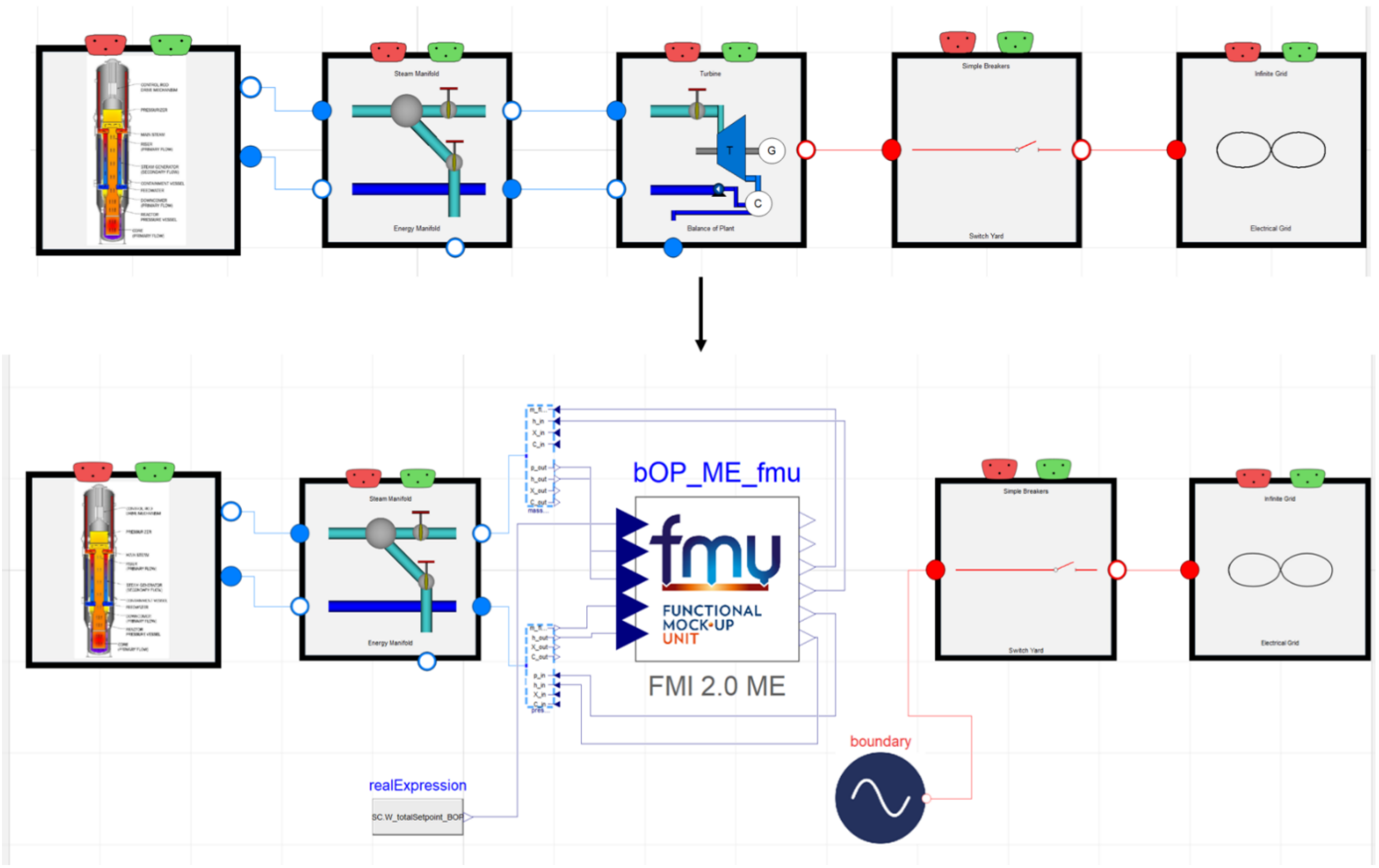


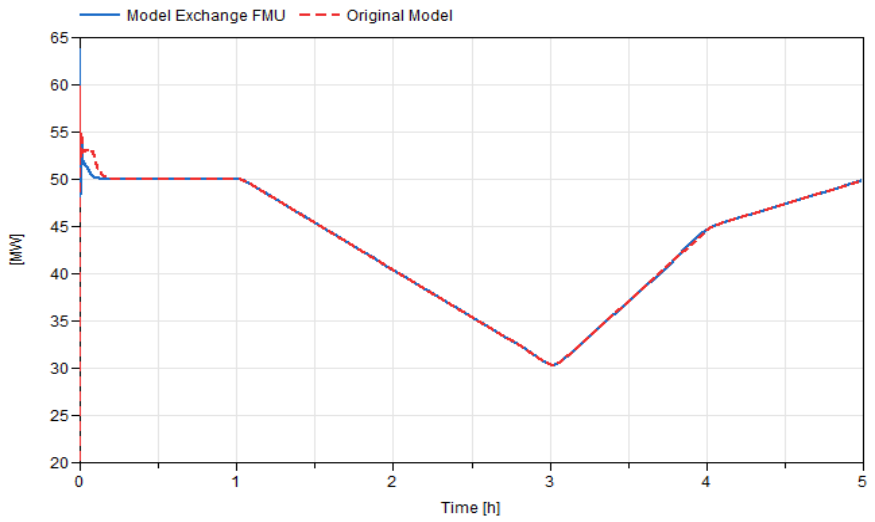
Supervisory Control/Control of Each Subsystem

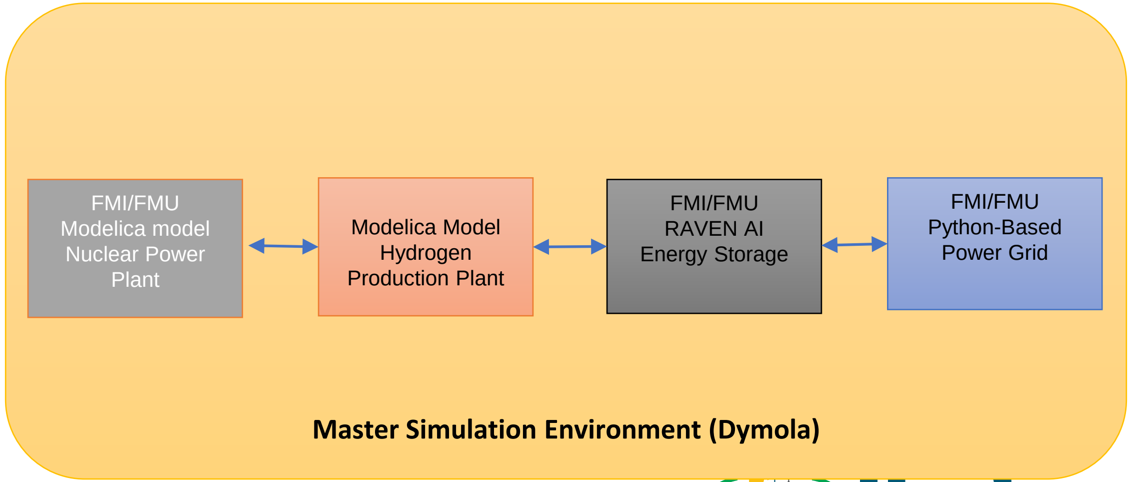
- Individual System level controllers remain in the Dymola/FMI components and subcomponents.
- A time dependent timeSeries.txt for load variations is given via an external .txt. File that is loaded into the Modelica system.
- This timeSeries.txt file is created by the RAVEN/HERON/Reference Governor workflow that is then sent Modelica.

IES
Integrated Energy Systems

FMI/FMU







Integrated Energy Systems

Available Literature on Models

- Literature:

1) https://www.osti.gov/biblio/1569288-status-report-nuscale-module-developed-modelica-framework. -- Frick, Konor L. Status Report on the NuScale Module Developed in the Modelica Framework. United States: N. p., 2019. Web. doi:10.2172/1569288.
2) https://www.osti.gov/biblio/1333156-status-component-models-developed-modelica-framework-high-temperature-steam-electrolysis-plant-gas-turbine-power-plant -- Suk Kim, Jong, McKellar, Michael, Bragg-Sitton, Shannon M., and Boardman, Richard D. Status on the Component Models Developed in the Modelica Framework: High-Temperature Steam Electrolysis Plant & Gas Turbine Power Plant. United States: N. p., 2016. Web. doi:10.2172/1333156.
3) https://www.osti.gov/biblio/1468648-status-report-component-models-developed-modelica-framework-reverse-osmosis-desalination-plant-thermal-energy-storage --Kim, Jong Suk, and Frick, Konor. Status Report on the Component Models Developed in the Modelica Framework: Reverse Osmosis Desalination Plant & Thermal Energy Storage. United States: N. p., 2018. Web. doi:10.2172/1468648.
4) https://www.osti.gov/biblio/1333156-status-component-models-developed-modelica-framework-high-temperature-steam-electrolysis-plant-gas-turbine-power-plant -- Suk Kim, Jong, McKellar, Michael, Bragg-Sitton, Shannon M., and Boardman, Richard D. Status on the Component Models Developed in the Modelica Framework: High-Temperature Steam Electrolysis Plant & Gas Turbine Power Plant. United States: N. p., 2016. Web. doi:10.2172/1333156
5) https://www.osti.gov/biblio/1557660-design-operation-sensible-heat-peaking-unit-small-modular-reactors -- Frick, Konor, Doster, Joseph Michael, and Bragg-Sitton, Shannon. Design and Operation of a Sensible Heat Peaking Unit for Small Modular Reactors. United States: N. p., 2018. Web. doi:10.1080/00295450.2018.1491181.
6) https://www.osti.gov/biblio/1557661-thermal-energy-storage-configurations-small-modular-reactor-load-shedding -- Frick, Konor, Misenheimer, Corey T., Doster, J. Michael, Terry, Stephen D., and Bragg-Sitton, Shannon. Thermal Energy Storage Configurations for Small Modular Reactor Load Shedding. United States: N. p., 2018. Web. doi:10.1080/00295450.2017.1420945.
7) https://www.osti.gov/biblio/1562960-dynamic-performance-analysis-high-temperature-steam-electrolysis-plant-integrated-within-nuclear-renewable-hybrid-energy-systems -- Kim, Jong Suk, Boardman, Richard D., and Bragg-Sitton, Shannon M. Dynamic performance analysis of a high-temperature steam electrolysis plant integrated within nuclear-renewable hybrid energy systems. United Kingdom: N. p., 2018. Web. doi:10.1016/j.apenergy.2018.07.060.
8) https://www.osti.gov/biblio/1357452-modeling-control-dynamic-performance-analysis-reverse-osmosis-desalination-plant-integrated-within-hybrid-energy-systems. Kim, Jong Suk, Chen, Jun, and Garcia, Humberto E. Modeling, control, and dynamic performance analysis of a reverse osmosis desalination plant integrated within hybrid energy systems. United States: N. p., 2016. Web. doi:10.1016/j.energy.2016.05.050.