INL/MIS-22-66314



# FORCE – Transient Physical Modeling Workshop

#### Hybrid for Analysts

Prepared and Presented by: Dr. Daniel Mikkelson and Dr. Konor Frick

And a state of the second state of the

# **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



# 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

3

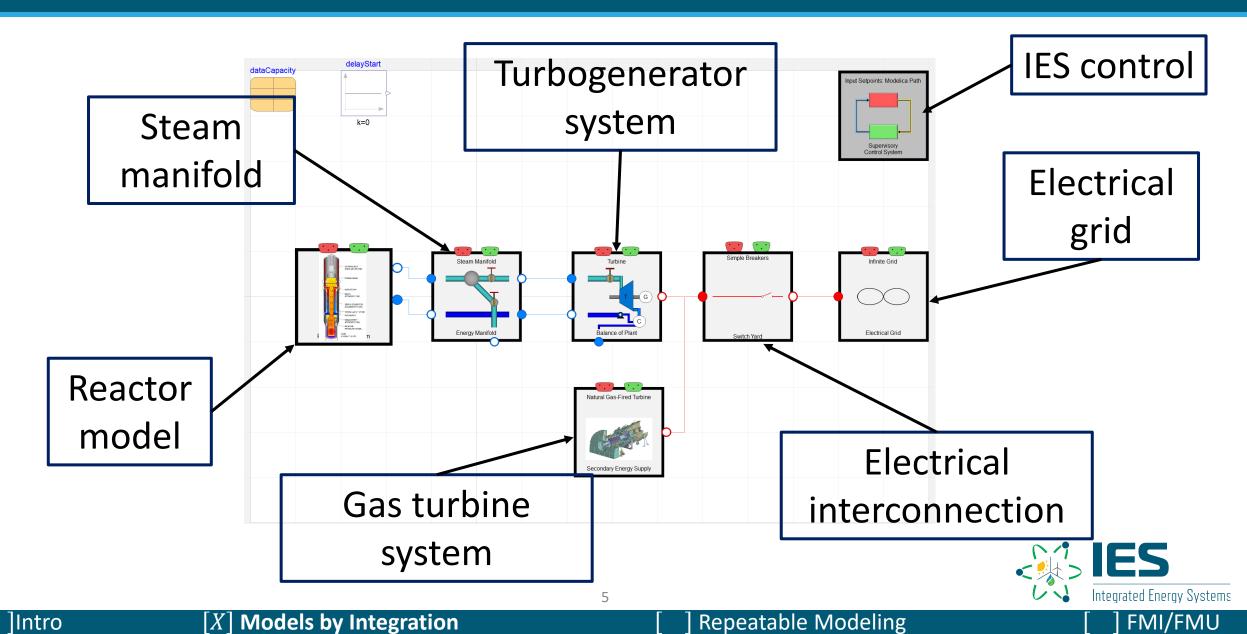


# Integration of Existing Models

- 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

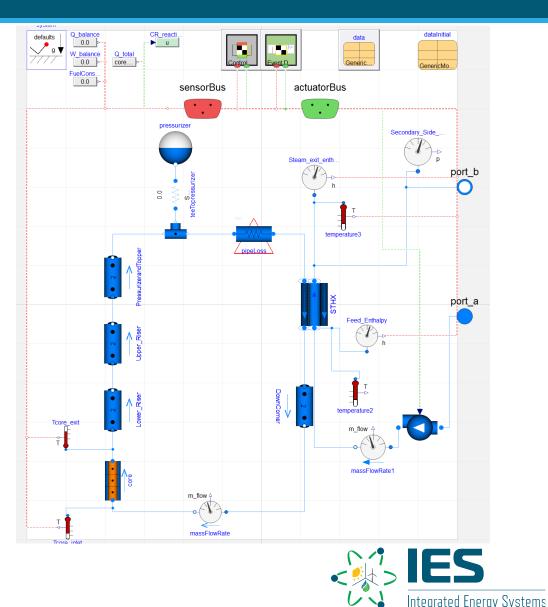


# **Example: IES**



## **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.

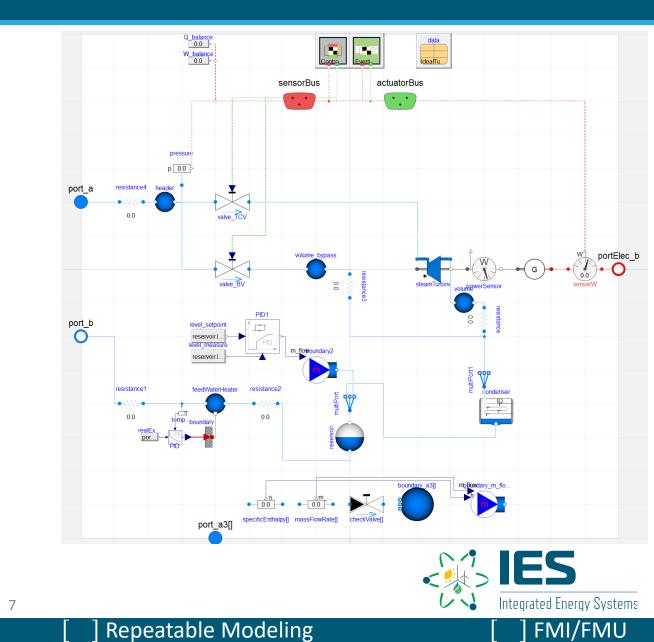


FMI/FMU

# **Example: IES – Turbogenerator**

- Turbogenerator system demonstrates five connection types
  - Fluid (blue)

- Heat (red solid)
- Mechanical (gray)
- Electrical (red solid)
- Control (red & green dashed)



# 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



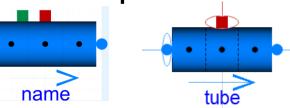
**Repeatable Modeling** 

# **Example: Shell & Tube Heat Exchanger**

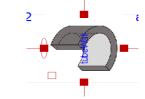
- What do we need to make a STHX?
  - Shell fluid flow path

Tube fluid flow path

name

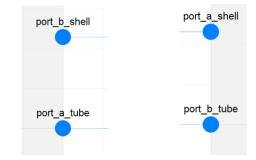


 Pipe model to establish conductivity



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

 External fluid connectors



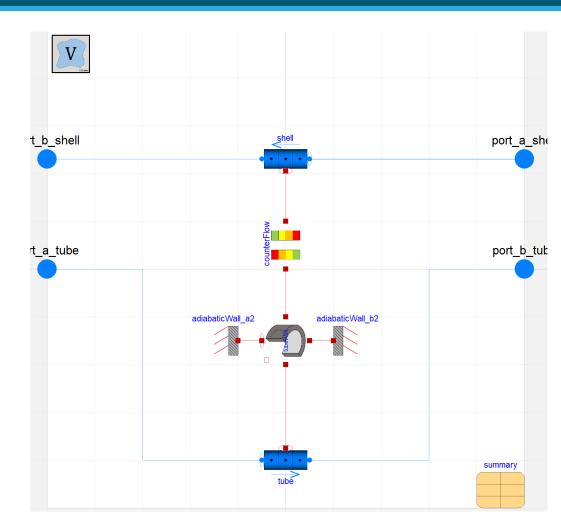
**Repeatable Modeling** 



[X] Models by Integration

# **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?

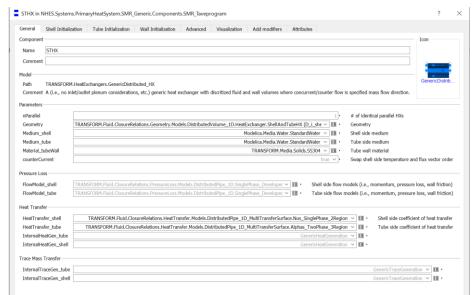




**Repeatable Modeling** 

## 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)



**Repeatable Modeling** 

### **Passing Parameters**

eneral Shell Initializ	ation Tube Initialization Wall Initialization Advanced Visualization Add modifiers Attributes	
omponent		Icon
Name STHX		
Comment		
odel		GenericDistrib
	I.HeatExchangers.GenericDistributed_HX let/outlet plenum considerations, etc.) generic heat exchanger with discritized fluid and wall volumes where concurrent/counter flow is specified mass flow direction.	
comment A (i.e., no in	reçolulet prenum consideradoris, etc.) generic neat exchanger with discritezer huid and wair volumes where concurreng counter now is specified mass now direction.	
arameters		
nParallel	≠ of identical parallel HXs	
Geometry	TRANSFORM.Fluid.ClosureRelations.Geometry.Models.DistributedVolume_1D.HeatExchanger.ShellAndTubeHX (D_i_st 🗠 🔢 · Geometry	
Medium_shell	Modelica.Media.Water.StandardWate V 🖩 Edit   Shell side medium	
Medium_tube	Modelica. Media. Water. Standard Water V 112 V 1	
Material_tubeWall counterCurrent	TRANSFORM.Media.Solids.SS304 ~> III → Tube wall material true ~> Swap shell side temperature and	
ressure Loss		
	TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D.SinglePhase_Developec        III · Shell side flow models (i.e., momentum, pressure)	loss, wall friction)
ressure Loss FlowModel_shell FlowModel_tube	TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D.SinglePhase_Developec >>       III +       Shell side flow models (i.e., momentum, pressure         TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D.SinglePhase_Developec >>       III +       Tube side flow models (i.e., momentum, pressure	
FlowModel_shell FlowModel_tube		
FlowModel_shell FlowModel_tube eat Transfer		loss, wall friction)
FlowModel_shell FlowModel_tube eat Transfer HeatTransfer_shell	TRANSFORM. Huid. ClosureRelations. PressureLoss. Models. DistributedPipe_1D. SinglePhase_Developec 💙 🏭 🔹 Tube side flow models (i.e., momentum, pressure	loss, wall friction)
FlowModel_shell FlowModel_tube eat Transfer HeatTransfer_shell HeatTransfer_tube	TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D.SinglePhase_Developec       III       Tube side flow models (i.e., momentum, pressure         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_2Region       III       Shell side coefficient	loss, wall friction)
FlowModel_shell FlowModel_tube	TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D.SinglePhase_Developec       III       Tube side flow models (i.e., momentum, pressure         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_2Region       III       Shell side coefficient         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Alphas_TwoPhase_3Region       III       Shell side coefficient	loss, wall friction)
FlowModel_shell FlowModel_tube leat Transfer HeatTransfer_shell HeatTransfer_tube InternalHeatGen_tube InternalHeatGen_shell	TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D.SinglePhase_Developec       III       Tube side flow models (i.e., momentum, pressure         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_2Region       III       Shell side coefficient         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Alphas_TwoPhase_3Region       III       Shell side coefficient         GenericHeatGeneration       III       Tube side coefficient       III       Tube side coefficient	loss, wall friction)
FlowModel_shell FlowModel_tube eat Transfer — HeatTransfer_shell HeatTransfer_tube InternalHeatGen_tube InternalHeatGen_shell race Mass Transfer —	TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D_SinglePhase_Developec       III       Tube side flow models (i.e., momentum, pressure         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_2Region       III       Shell side coefficie         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_3Region       III       Shell side coefficie         GenericHeatGeneration       III       GenericHeatGeneration       III       Tube side coefficie	loss, wall friction) nt of heat transfer nt of heat transfer
FlowModel_shell FlowModel_tube leat Transfer HeatTransfer_shell HeatTransfer_tube InternalHeatGen_tube InternalHeatGen_shell race Mass Transfer InternalTraceGen_tube	TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D_SinglePhase_Developec       III       Tube side flow models (i.e., momentum, pressure         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_ZRegion       III       Shell side coefficie         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_ZRegion       III       Shell side coefficie         GenericHeatGeneration       III       Coefficie       GenericHeatGeneration       III         GenericHeatGeneration       III       GenericTraceGeneration       III       GenericTraceGeneration	loss, wall friction) nt of heat transfer nt of heat transfer
FlowModel_shell FlowModel_tube leat Transfer HeatTransfer_shell HeatTransfer_tube InternalHeatGen_tube InternalHeatGen_shell race Mass Transfer InternalTraceGen_tube	TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D_SinglePhase_Developec       III       Tube side flow models (i.e., momentum, pressure         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_2Region       III       Shell side coefficie         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_3Region       III       Shell side coefficie         GenericHeatGeneration       III       GenericHeatGeneration       III       Tube side coefficie	loss, wall friction) nt of heat transfer nt of heat transfer
FlowModel_shell FlowModel_tube eat Transfer — HeatTransfer_shell HeatTransfer_tube InternalHeatGen_tube InternalHeatGen_shell race Mass Transfer — InternalTraceGen_tube	TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D_SinglePhase_Developec       III       Tube side flow models (i.e., momentum, pressure         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_ZRegion       III       Shell side coefficie         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_ZRegion       III       Shell side coefficie         GenericHeatGeneration       III       Coefficie       GenericHeatGeneration       III         GenericHeatGeneration       III       GenericTraceGeneration       III       GenericTraceGeneration	loss, wall friction) nt of heat transfer nt of heat transfer
FlowModel_tube Heat Transfer	TRANSFORM.Fluid.ClosureRelations.PressureLoss.Models.DistributedPipe_1D_SinglePhase_Developec       III       Tube side flow models (i.e., momentum, pressure         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_ZRegion       III       Shell side coefficie         TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_ZRegion       III       Shell side coefficie         GenericHeatGeneration       III       Coefficie       GenericHeatGeneration       III         GenericHeatGeneration       III       GenericTraceGeneration       III       GenericTraceGeneration	loss, wall friction) nt of heat transfer nt of heat transfer ion ∨ 夏夏, ion ∨ 夏夏,

	data.nTubes_steamGenerator    # of tubes per heat exchanger				
nR	2 Number of radial nodes in wall (r-direction)				
nSurfaces_tube	1 · Number of transfer (heat/mass) surfaces				
puts Elevation —					
height_a_tube	0 M Elevation at port_a: Reference value only. No impact on calculations.				
height_b_tube	height_a_tube + sum(dheights_tube) • m Elevation at port_b: Reference value only. No impact on calculations.				
angle_tube	○ · · · · · · · · · · · · · · · · · · ·				
dheight_tube	length_tube*sin(angle_tube) • m Height(port_b) - Height(port_a) distributed by flow segment				
nputs: Tube Wall —					
drs	fill(th_wall/nR, nR, nV) III • m Tube unit volume lengths of r-dimension				
th_wall	data.th_steamGenerator_tube + m Tube wall thickness				
nputs					
	data.d_steamGenerator_tube_inner > m Characteristic dimension (e.g., hydraulic diameter)				
dimension_tube	dua.u_steamoenerator_tabe_inner				
-	0.25*pi*dimension_tube*dimension_tube + m <sup>2</sup> Cross-sectional flow areas				
crossArea_tube					
crossArea_tube perimeter_tube	0.25*pi*dimension_tube*dimension_tube m <sup>2</sup> Cross-sectional flow areas				
dimension_tube crossArea_tube perimeter_tube length_tube roughness_tube	0.25*pi*dimension_tube*dimension_tube       m²       Cross-sectional flow areas         4*crossArea_tube/dimension_tube       m       Wetted perimeters				



Intro

#### ] Repeatable Modeling

# 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

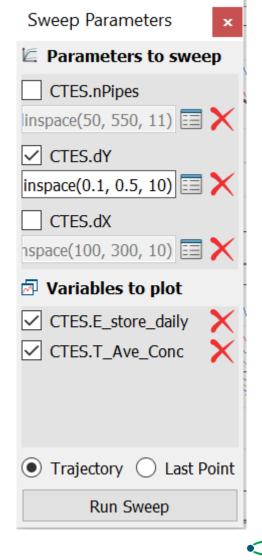


FMI/FMU

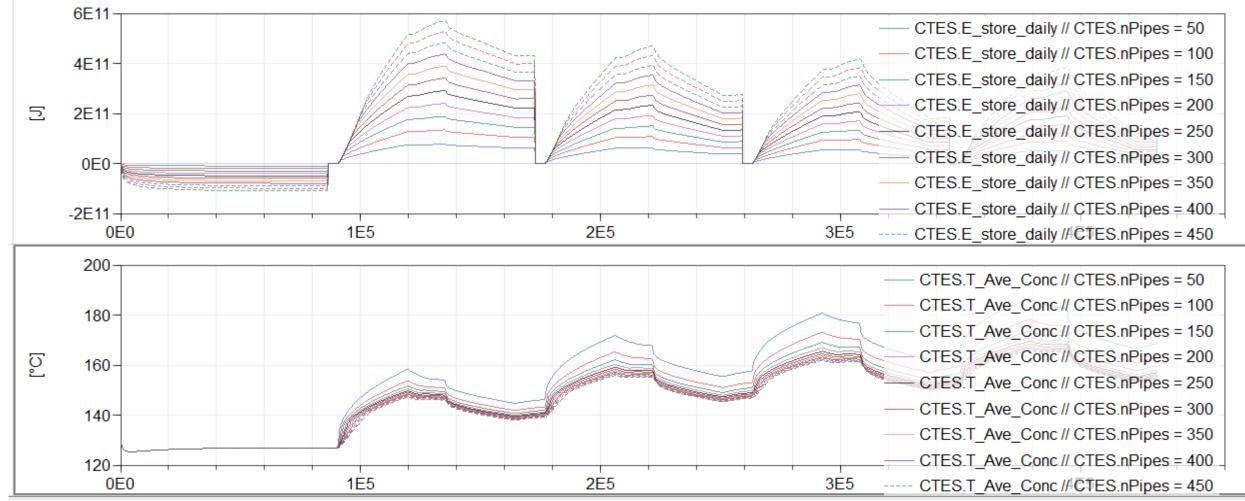
**Repeatable Modeling** 

# **Parameter Sweeping**

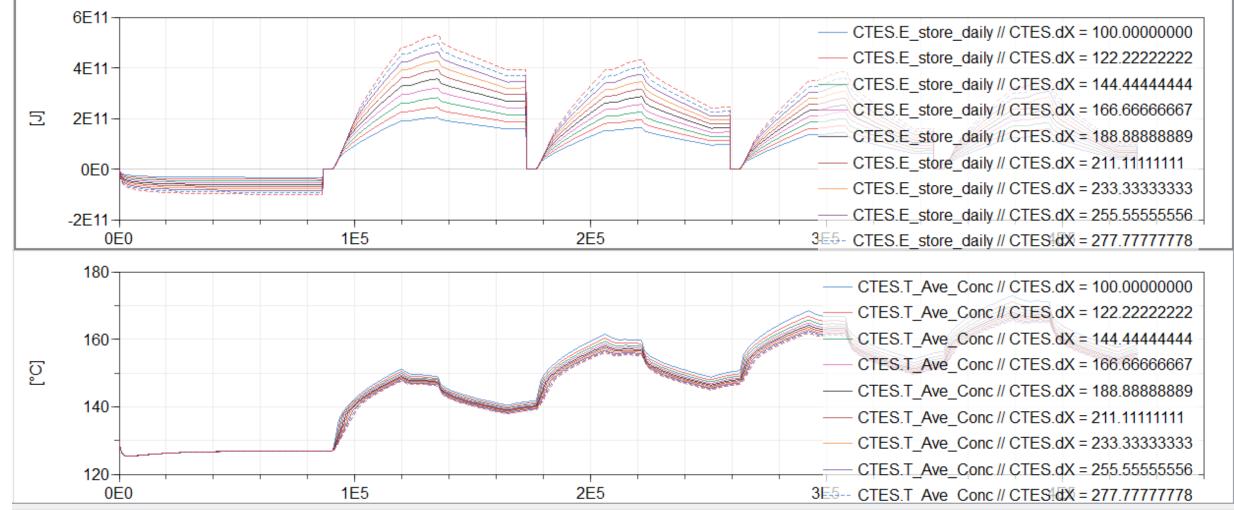
- Dymola has internal parameter sweeping method
- Allows for output space generation across single altered parameter at a time
- Auto generates separate output files to keep post simulation
- Auto generates plotting set of output values desired by user



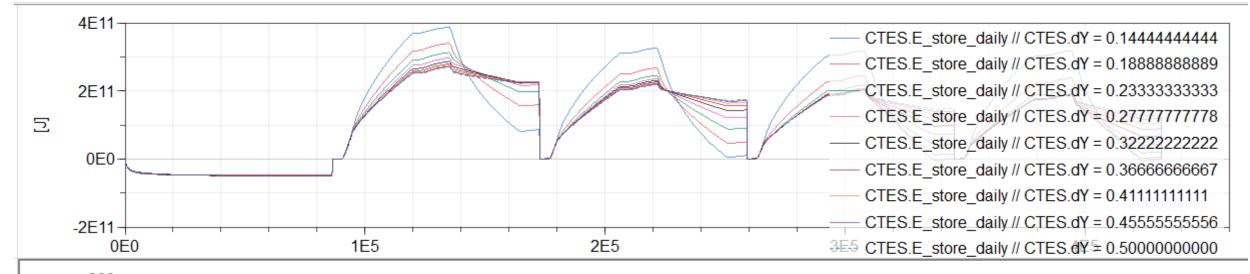


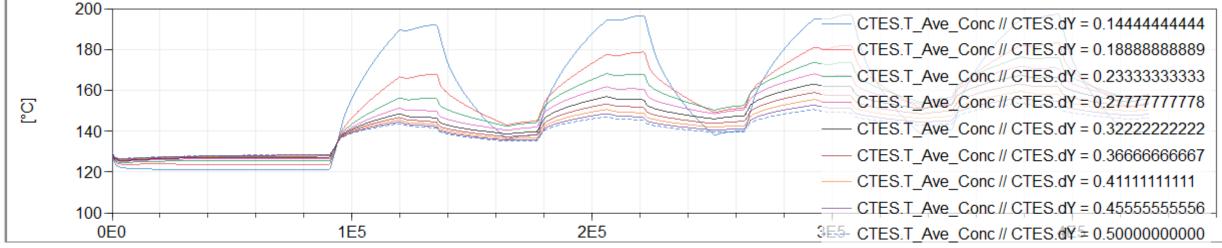














## **Manual Parameter Sweep**

- 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



- 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

General Add modifiers Attri	butes	
Component	Ico	n
Name dataInitial		
Comment		atalnitial NS
comment		
lodel		Generic
Path NHES.Systems.PrimaryH Comment	leatSystem.SMR_Generic.Components.Data.DataInitial_NS	Genenc
Parameters		
d_start_core_coolantSubchannel	{0.72999456787.0.70768652344.0.68389465332.0.65836	a/cm³
p_start_core_coolantSubchannel	{12903247.0,12898190.0,12893307.0,12888614.0}	100 million
T_start_core_coolantSubchannel	{540.2313,558.79364,576.1813,586.9483}	
h start core coolantSubchannel	{1317572.25,1374710.125,1431848.125,1488985.875}	
d_start_hotLeg	{0.6582980957,0.65820166016}	
p_start_hotLeg		bar
T_start_hotLeg	{324.70516357,324.67647705}	
h_start_hotLeg	{1488930.125,1488878.625}	
d_start_coldLeg	{0.75105932617}	
p_start_coldLeg	{129.36737}	
T_start_coldLeg	{285.19472656} II	
h_start_coldLeg		J/kg
d_start_STHX_tube	{1200+3+.123} = { {0.80191125488,0.11777983093,0.06753452301,0.04695 ]	
p_start_STHX_tube	{39.281535,39.2524275,39.1300875,38.91694,38.610355	
T_start_STHX_tube	{248.02053223,249.22127686,249.04030762,248.721887	
h_start_STHX_tube		
d_start_STHX_shell	2374421.2,2749133.0,2919464.0,2980839.5,3004198.5} (0.66202679443,0.66976708984,0.68217218018,0.69354	
p_start_STHX_shell	{128.10909.128.16239.128.19835.128.23497.128.27221.	
T_start_STHX_shell	01654,547.5736,541.4915,535.5601,527.6039,516.3904}	
h_start_STHX_shell	{1480592.25,1463564.625,1435439.875,1408754.25,138	
d_start_inletPlenum	0.751037	
p_start_inletPlenum	129.207	
T_start_inletPlenum	285.193	
h_start_inletPlenum	1.26043e+06	
d_start_outletPlenum	0.658335	
p_start_outletPlenum	128.884	
T_start_outletPlenum	324.724	
h_start_outletPlenum		
	326, 753.8575; 620.802, 676.6141, 692.713, 665.06805]	
Ts_start_core_fuelModel_region_2	629.82477; 547.5703, 569.67633, 586.9591, 594.09796]	
	5, 590.4041; 540.2313, 558.79364, 576.1813, 586.9483]	
T_start_STHX_tubeWall	, 554.01654, 560.9657, 576.07135, 583.2033, 585.6959]	
p_start_pressurizer	12807852	
level_start_pressurizer	1.18567	
h_start_pressurizer	1.47822e+06	
d_start_pressurizer_tee	0.658202	-
p_start_pressurizer_tee	12807852 •	
T_start_pressurizer_tee	586.90674	ĸ



- 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

Save Start Values in Model		?	×
Source for start values			
<ul> <li>Current Variable Browser content</li> <li>Initialize the model and save the results</li> </ul>			
Store options			
<ul> <li>Store values in current model</li> <li>Store values in new model</li> </ul>			
Name:			
SMR_IES_CTES			
Description:			
Extends:			
NHES.Systems.Examples.SMR_IES_CTES			
Insert in package:			
NHES.Systems.Examples		✓ <sup>+0</sup> / <sub>0</sub>	, 🗋
Open new class in:			
This tab			$\sim$
Advanced options for storing start guesses			
Save changes in parameters and in initial values of states			
Overwrite parametrized start attributes for below selection			
Additionally, save changes in the start attributes of:			
<ul> <li>Iteration variables</li> <li>Iteration variables and torn variables</li> <li>Outputs, auxiliary variables, and states</li> </ul>			
Only save start guesses for additional variables at start time. Other useage may cause unwanted changes in the model parametrization. These advanced options are only intended for saving start guesses and must not be used to continue simulations from times later than the start time.			
Advanced <<	ОК	Can	cel



#### X Repeatable Modeling

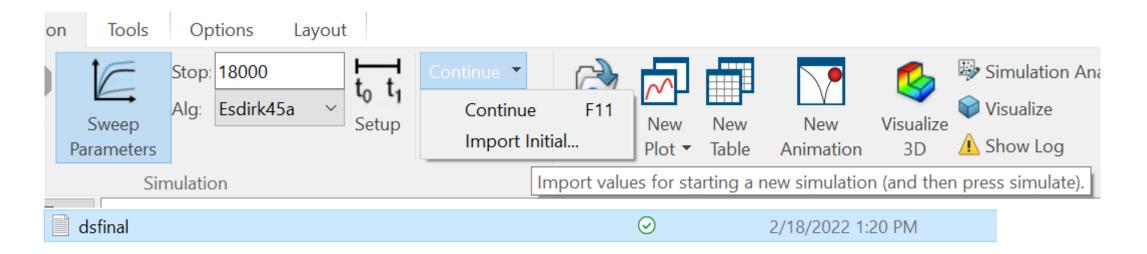
```
DUP
 CS (
  BV_openingNominal(k(start=0.001)),
  PID_BV_opening(
    I(k(start=0.1)),
                                               actuatorBus(opening BV(start=0.001), opening TCV(start=0.5)),
    addP(
     k1(start=1.0),
                                               boundary(port(T(start=421.1691093331664))),
     u1(start=0.030000001485),
     u2(start=0.030000001485)),
                                               boundary2 (medium (
    gainPID(k(start=-1.0)),
    gainTrack(k(start=-1.1111111111111111))),
    gain_u_m(k(start=5E-10)),
                                                     T(start=298.16763607879363),
    gain_u_s(k(start=5E-10)),
    limiter(uMax(start=0.999), uMin(start=-0.0009)),
                                                     T degC(start=25.017636078793657),
    null bias(k(start=0.0)),
    vMax(start=0.999),
                                                     d(start=998.544943058541),
    yMin(start=-0.0009)),
  PID TCV opening(
                                                     p bar(start=34.47380000000004),
    I(k(start=2.0)),
    addP(
     k1(start=1.0),
                                                      sat(Tsat(start=514.8425665422984)),
     u1(start=0.0086185),
     u2(start=0.0086185)),
                                                      u(start=104571.53362749857)), ports(h outflow(start={108023.9370710939}),
    gainTrack(k(start=1.1111111111111111))),
    gain u m(k(start=2.5E-09)),
                                                         - / - + - - + - ( 2//7200 01) ) )
    gain u s(k(start=2.5E-09)),
    limiter(uMax(start=0.5), uMin(start=-0.4999)),
                                                       Kt(start=0.013324090093760938),
    null_bias(k(start=0.0)),
    u s(start=3447400.0),
                                                       Q mech(start=65056859.70577499),
    yMax(start=0.5),
    vMin(start=-0.4999)).
                                                       Q units(start={42261285.79911617,42261285.79911617}),
  TCV openingNominal(k(start=0.5)),
  delayStartBV(start=100.0),
                                                       Q units start(start={42261285.79911617,42261285.79911617}),
  p Nominal1(k(start=3447400.0)),
   switch P setpoint(y(start=60000000.297)),
                                                       Obs(start={-9732855.946228676,-9732855.946228676}),
  valvedelav(k(start=100.0)),
  valvedelayBV(k(start=100.0))),
                                                       T a start(start=293.15),
 PID(
  I(k(start=2.0)),
                                                       T b start(start=293.15),
  addP(
                                                       T nominal(start=293.15),
    k1(start=1.0),
    u1(start=1.0),
                                                       bubble in(d(start=820.3581983078773), h(start=1013666.6724914373)
    u2(start=1.0)),
  gainPID(k(start=100000000.0)),
                                                       bubble out(d(start=989.8436373961912), h(start=191812.29519356362
  gainTrack(k(start=1.11111111111111112E-08)),
  gain_u_m(k(start=0.002374343174368348)),
                                                       d nominal(start=13.671247252758716),
  gain_u_s(k(start=0.002374343174368348)),
  k m(start=0.002374343174368348),
                                                       dew in(d(start=15.307197090608243), h(start=2803284.170249812)),
   k s(start=0.002374343174368348),
  limiter(
                                                       dew out(d(start=0.06816373081854721), h(start=2583886.8570257137)
    u(start=100000000.0),
    uMax(start=1E+60),
                                                       h a start(start=2997670.0),
    uMin(start=-1E+60)),
  null bias(k(start=100000000.0)),
                                                       h b start(start=2058530.3155751962),
  u_m(start=421.1691093331664),
   yMin(start=-1E+60)),
                                                       h is(start=2070197.5860370956),
 PID1(
  PID(
                                                       h out(start=2209318.4481315315),
    I(k(start=2.0)),
    Nd(start=10.0),
                                                       p a start(start=3337380.0),
    Ni(start=0.9),
                                                       p b start(start=10000.0),
    Td(start=0.1),
    Ti(start=0.5),
                                                       p inlet nominal(start=3337380.0),
    addP(
     k1(start=1.0),
                                                       p outlet nominal(start=10000.0),
     u1(start=1.0),
     u2(start=1.0)),
                                                       p ratio(start=0.0032668200354825697),
    gainPID(k(start=100.0)),
    gainTrack(k(start=0.011111111111111111))),
                                                       portHP(
    gain u m(k(start=0, 1))
                                                                                                                                                                          Integrated Energy Systems
```

Intro

[X] Repeatable Modeling

FMI/FMU

 Another method of importing initial conditions is using default output format



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



# **RAVEN Interfacing**

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

```
-2 1.000000000000001E-01
    280 # nuScale Tave enthalpy Pressurizer CR.PID.k
1
-2 5.000000000000000E-01 9.999999999999997E-61 1.000000000000000E+100
    280 # nuScale Tave enthalpy Pressurizer CR.PID.Ti
1
-2 1.000000000000001E-01
                                                   1.0000000000000000E+100
    280 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.Td
1
-2
        0
                                                         0
    280 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.yb
1
   5.000000000000000E-01
-2
                                                         0
    256 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.k_s
6
   5.000000000000000E-01
-2
                                                         0
    256 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.k_m
6
   5.66000000000001E+00
-2
                                                         0
    280 # nuScale_Tave_enthalpy_Pressurizer_CR.PID.yMax
1
-2
        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 "Dymolalnitialisation"



# **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**

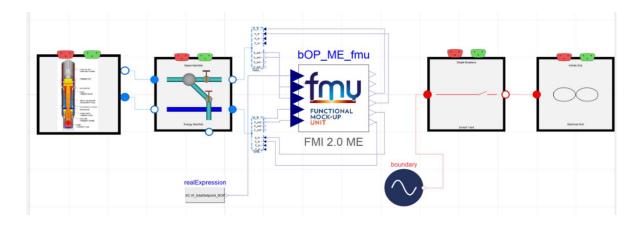


# **Functional Mockup Units**

- 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.



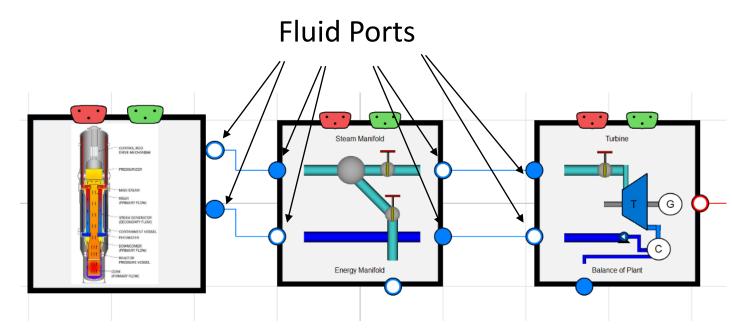
Picture from: https://fmi-standard.org/





### **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.</li>

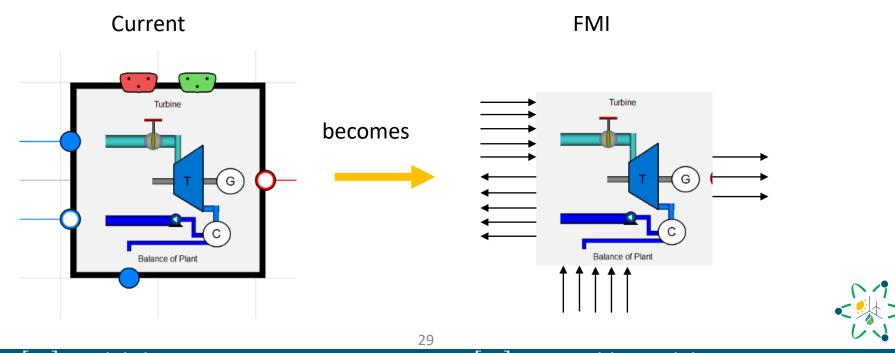


**Repeatable Modeling** 



### **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.



Integrated Energy Systems

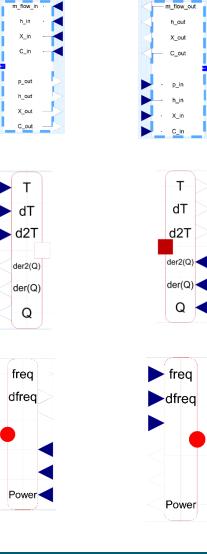
[X] FMI/FMU

### **Standardized FMI/FMU Adaptors**

Fluid Adaptors

Heat Adaptors

**Electrical Adaptors** 



p\_in

h\_in

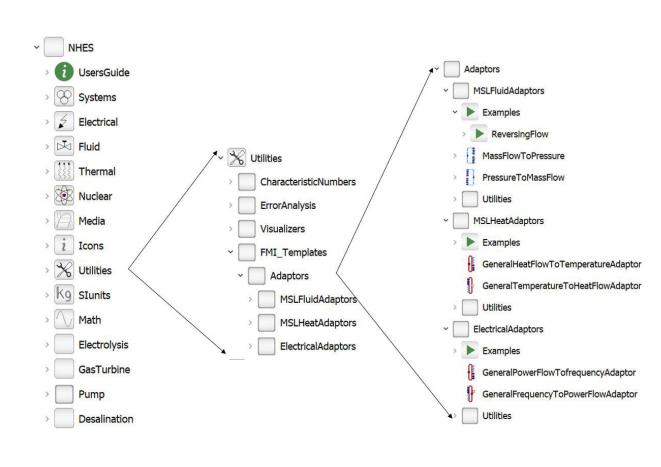
X\_in

C\_in

Т

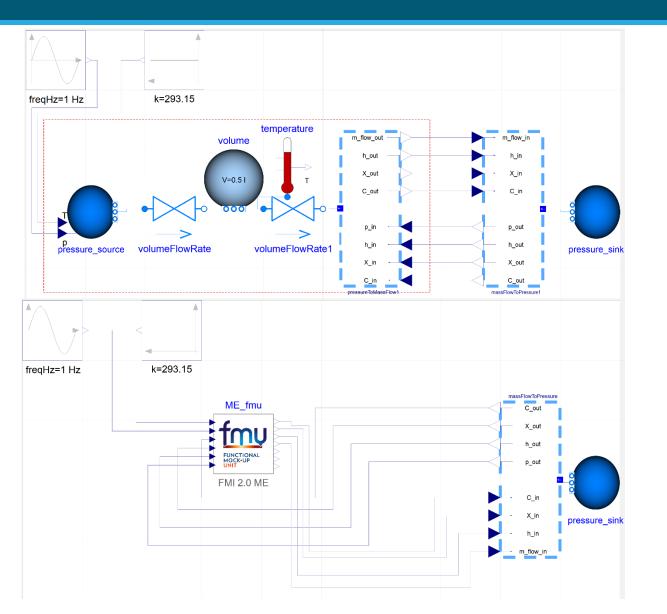
dT

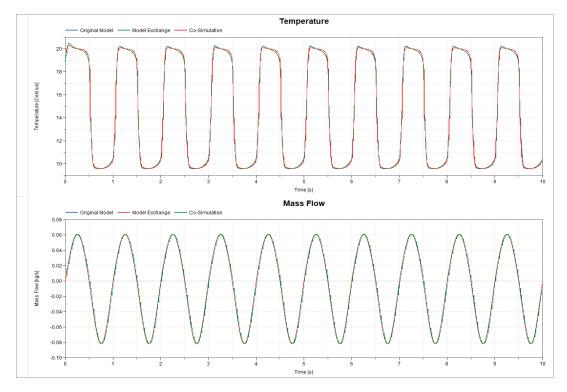
Q <





#### **FMI/FMU** Demonstration







#### [ ] Models by Integration

Intro

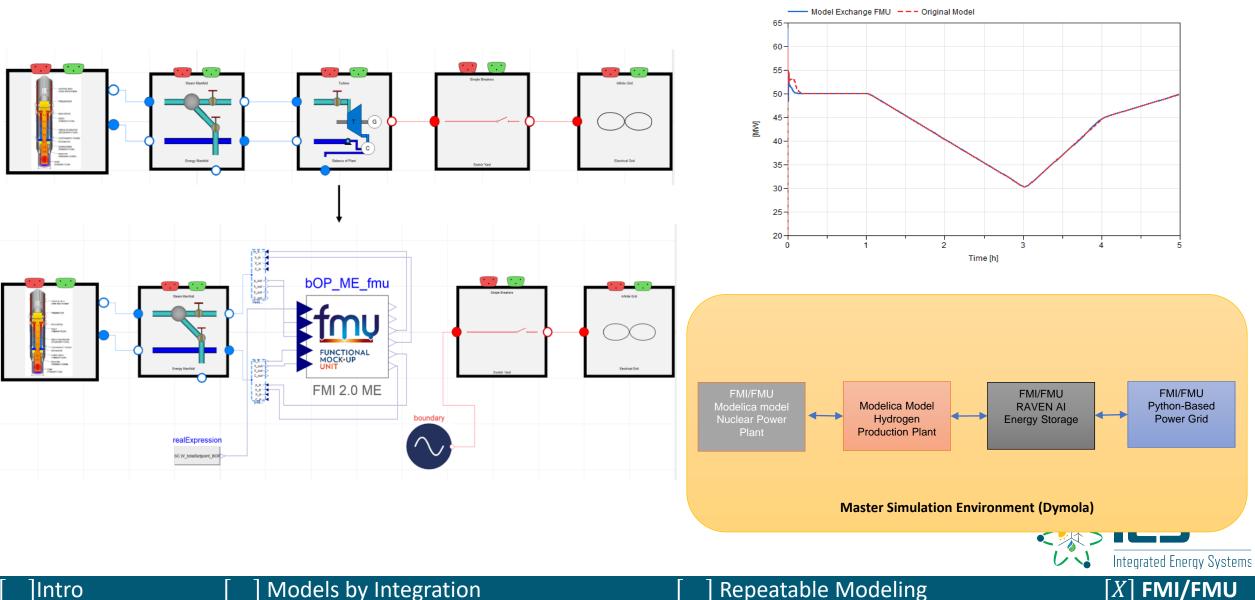
#### ] Repeatable Modeling

### 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.



**Repeatable Modeling** 



Intro

#### Models by Integration

#### **Available Literature on Models**

- Literature:
  - 1) <u>https://www.osti.gov/biblio/1569288-status-report-nuscale-module-developed-modelica-framework</u>. -- Frick, Konor L. Status Report on the NuScale Module Developed in the Modelica Framework. United States: N. p., 2019. Web. doi:10.2172/1569288.
  - <u>https://www.osti.gov/biblio/1333156-status-component-models-developed-modelica-framework-high-temperature-steam-electrolysis-plant-gas-turbine-power-plant</u> -- 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.
  - <u>https://www.osti.gov/biblio/1468648-status-report-component-models-developed-modelica-framework-reverse-osmosis-desalination-plant-thermal-energy-storage</u> --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) <u>https://www.osti.gov/biblio/1333156-status-component-models-developed-modelica-framework-high-temperature-steam-electrolysis-plant-gas-turbine-power-plant</u> -- 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) <u>https://www.osti.gov/biblio/1557660-design-operation-sensible-heat-peaking-unit-small-modular-reactors</u> -- 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) <u>https://www.osti.gov/biblio/1557661-thermal-energy-storage-configurations-small-modular-reactor-load-shedding</u> -- 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) <u>https://www.osti.gov/biblio/1562960-dynamic-performance-analysis-high-temperature-steam-electrolysis-plant-integrated-within-nuclear-renewable-hybridenergy-systems</u> -- 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.
  - <u>https://www.osti.gov/biblio/1357452-modeling-control-dynamic-performance-analysis-reverse-osmosis-desalination-plant-integrated-within-hybrid-energy-systems</u>. 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.