



# Status Report on IES Plug-and-Play Framework

November 2020

Konor L. Frick  
Andrea Alfonsi  
Cristian Rabiti



*INL is a U.S. Department of Energy National Laboratory  
operated by Batelle Energy Alliance, LLC*

#### **DISCLAIMER**

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

# **Status Report on IES Plug-and-Play Framework**

**Konor L. Frick  
Andrea Alfonsi  
Cristian Rabiti**

**November 2020**

**Idaho National Laboratory  
INL IES Program  
Idaho Falls, Idaho 83415**

**<http://www.inl.gov>**

**Prepared for the  
U.S. Department of Energy  
Office of Nuclear Energy  
Under DOE Idaho Operations Office  
Contract DE-AC07-05ID14517**

*Page intentionally left blank*

## **ABSTRACT**

This report discusses the status of the flexible plug-and-play framework development currently ongoing that aims to integrate Modelica/Dymola with the Risk Analysis and Virtual ENvironment (RAVEN) software in terms of both Functional Mock-Up Interface (FMI)/Functional Mock-Up Unit (FMU) construction and repository structures that aim to ease the sharing and simulation of complex dynamic models.

This report discusses the FMI/FMU adaptors that have been created within the HYBRID repository to allow users to quickly export models, such as FMUs. Several examples are shown that highlight the step-by-step process of converting an existing Modelica model into an FMU for use within the Dymola platform. Simulation results demonstrate that, while minor differences may occur, the overall control, trends, and solution integrity are maintained between standard Modelica simulation and FMU simulation results. However, it is worth noting that, for small systems, the FMU results have a slower simulation time than the Modelica only simulation.

Using this process, a company can provide models that contain proprietary information to entities without disclosing any of the information about the model that could be considered business sensitive. Such an ability would allow institutions to bypass the necessity of “whitewashing” data.

In addition to the investigative work being conducted on FMUs and FMIs, a series of updates to the hybrid repository has been completed. These updates include the addition of Modelica system-level regression tests and software quality assurance documentation that ensure that modifications to the Modelica models do not alter system-level model results.

Overall, extensive work has been completed on developing FMUs and FMIs from existing models, understanding the requirements and limitations of FMUs, and open-sourcing the HYBRID repository with an integrated regression system.

*Page intentionally left blank*

# CONTENTS

ACRONYMS .....	ix
1. INTRODUCTION.....	1
2. MODELICA TO FMU ADAPTATION .....	1
2.1 Adaptors .....	1
2.1.1 Fluid Port Adaptors .....	3
2.1.2 Thermal Port Adaptors.....	7
2.1.3 Electrical Port Adaptors .....	12
2.2 Functional Mock-Up Unit Options .....	12
2.3 Full-Scale IES Model.....	13
3. HYBRID REPOSITORY .....	16
4. IMPLEMENTATION IN RAVEN .....	20
5. ACCELERATION OF FMI AND FMU MODELS VIA ARTIFICIAL INTELLIGENCE .....	23
6. CONCLUSION .....	24
7. REFERENCES.....	25

# FIGURES

Figure 1. Fluid ports.....	2
Figure 2. Transition from a Modelica physical model to an FMI or FMU.....	2
Figure 3. FMU Template folder location within the larger NHES folder. ....	3
Figure 4. (Left) pressuretoMassFlow adaptor created by Modelon for use in the INL plug-and-play framework. This adaptor is best connected to a resistance port able to set the output mass flow rate. (Right) massFlowtoPressure adaptor created by Modelon for use in the INL plug-and-play framework. This adaptor is best connected to a volume port able to set the output pressure. ....	4
Figure 5. An example using adaptors that includes two pressure sources using moist air, one of which oscillates in pressure, causing a mass flow reversal. The red boxed in unit will become an FMU. ....	5
Figure 6. Example of a reversible flow using two pressures sources and moist air with a model exchange FMU.....	5
Figure 7. Example of a reversible flow using two pressures sources and moist air with a cosimulation FMU. ....	6

Figure 8. Mass flow to the pressure sink comparison across the original model, model exchange FMU, and cosimulation FMU (timestep = 0.02 seconds).....	6
Figure 9. (Left) GeneralTemperatureToHeatFlow adaptor for use in the INL plug-and-play framework. (Right) GeneralHeatFlowToTemperature adaptor for use in the INL plug-and-play framework. ....	7
Figure 10. Example meant to demonstrate the FMU variants available with the thermal FMU adaptors. The upper part demonstrates how to export two heat capacitors and connect them together in a target system. The lower part demonstrates how to export a conduction element that only needs temperatures for its conduction law and connects this conduction law in a target system between two capacitors. ....	8
Figure 11. Demonstration of an FMU variant example using model exchange for thermal heat port adaptors. ....	9
Figure 12. Collapse of the upper part into a single FMU for cosimulation. This is required because the frequency between the direct and inverse conduction problem is so fast that a single cut between the two could not be made without instabilities occurring. ....	9
Figure 13. Upper model connected with the combined direct, inverse cosimulation FMU. ....	10
Figure 14. Lower model cosimulation FMU. ....	10
Figure 15. Direct/inverse simulation results for the original, model exchange, and cosimulation (communication interval 0.002 seconds) FMU runs. (Top) Heat flow into the capacitor. (Bottom) Capacitor 3b temperature. ....	11
Figure 16. Conduction (lower model) simulation results for the original, model exchange, and cosimulation (communication interval 0.002 seconds) FMU runs. (Top) Heat flow into the capacitor. (Bottom) Capacitor 3b temperature.....	11
Figure 17. (Left) GeneralFrequencyToPowerFlow adaptor for use in the INL plug-and-play framework. (Right) GeneralPowerFlowTofrequency adaptor for use in the INL plug-and-play framework.....	12
Figure 18. FMI and FMU export options from Dymola2020FD01.....	13
Figure 19. Translation of Modelica Turbine Generator Model into an FMU ready design. ....	14
Figure 20. Transition from a Modelica model to an input-based Modelica model to an FMI-based simulation. ....	15
Figure 21. Comparison of turbine output results between the original model and model exchange FMU.....	15
Figure 22. New structure of the repository. ....	16



Figure 23. An example of tests run in the ROOK regression system. ....	18
Figure 24. Status of the required SQA documentation for the HYBRID modeling repository. ....	20
Figure 25. External model API. ....	21
Figure 26. FMI and FMU cosimulation protocol coupled with RAVEN. ....	22
Figure 27. FMI and FMU model exchange protocol coupled with RAVEN. ....	22
Figure 28. RAVEN hybrid model scheme. ....	23

## TABLES

Table 1. Synopsis of Modelica test cases. ....	17
--	----

*Page intentionally left blank*

## ACRONYMS

AI	artificial intelligence
API	application program interface
CPU	central processing unit
FMI	Functional Mock-Up Interface
FMU	Functional Mock-Up Unit
IES	Integrated Energy Systems
NHES	Nuclear Hybrid Energy Systems
RAVEN	Risk Analysis and Virtual ENvironment
SQA	software quality assurance

*Page intentionally left blank*

# **Status on the Development of the Infrastructure for a Flexible Modelica/RAVEN Framework for IES**

## **1. INTRODUCTION**

The Integrated Energy System (IES) Program is continuing its effort to design, improve, and enhance the modeling and simulation capabilities aimed at assessing the economic viability of integrated energy systems. Among the different activities, a key component of the recent research efforts is about the creation of an infrastructure for the flexible interaction of Modelica/Dymola models [1] with the Risk Analysis and Virtual ENvironment (RAVEN) ecosystem [2],[3].

In the past year, two additional status reports [4],[5] have been delivered with the initial description of the effort to create a “plug-and-play” repository of process models using Modelica, Functional Mock-Up Interfaces (FMIs), and Functional Mock-Up Units (FMUs) [6].

The goal of this document is to report the advancements achieved in the deployment of this research task.

## **2. MODELICA TO FMU ADAPTATION**

Modelica is a physical modeling language that relies on an acausal assignment of equations rather than a standard causal assignment. This means that an equation can only appear once and that the translator and system solvers will determine the proper way to assign the flow of information. In addition, since Modelica is a physical modeling language, there are the assignments of special variable containers “flow” and “stream” that have an inherent physical representation in the code. Flow variables have a direction and must sum to zero in a “connection.” The “stream” qualifier is used to qualify when a given element in a connection has an intensive property of flowing through a connector. These “connectors” include a singular flow variable with several stream variables alongside it. For example, a “fluid port” is a connector that has the mass flow rate as the “flow” variable and enthalpy as the “stream” variable. Mass flow is what physically goes through the connector, while enthalpy is a property of the mass flow. This nuance in variable types is particularly important when considering the translation of Modelica models into FMIs and FMUs. FMIs are only able to import and export real input and output signals. These signals are unable to retain the physical properties seen in Modelica and thus require special adaptors to translate them back into physical values for use in other Modelica models.

### **2.1 Adaptors**

For a connection between FMI’s and other Modelica models within the Dymola platform, a set of standardized variables and adaptors are needed to properly transmit energy values among subsystems. This is particularly true if the interconnection is between two physical models, such as a nuclear power plant and a turbine. This is because the physical models contain “ports,” as shown in Figure 1.

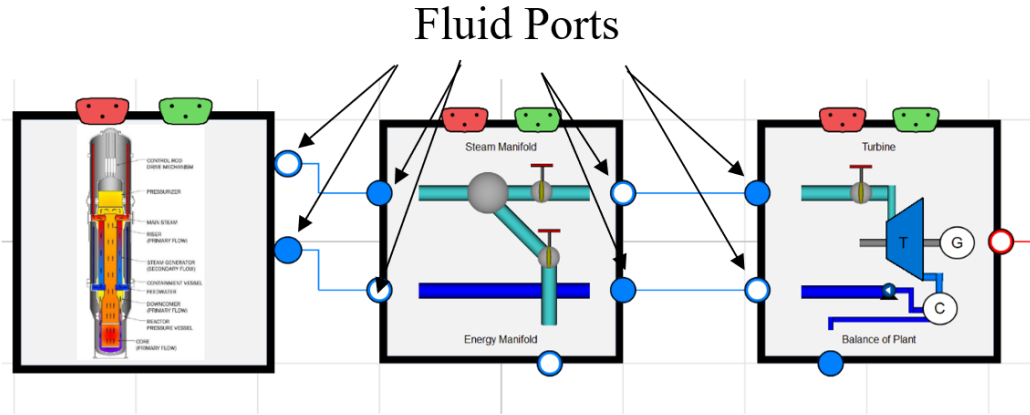


Figure 1. Fluid ports.

Each fluid port contains:

- Mass flow (flow variable),  $m\_flow$
- Conditional enthalpy (**stream** variable),  $h\_outflow$
- Pressure,  $P$
- Trace substance fraction (**stream** variable),  $C_i$
- Mass fraction (**stream** variable),  $X_i$ .

To properly transition from ports to input and output signals, the individual components of the ports must be broken out and assumed to be an input or an output. This is illustrated in Figure 2, where each of the fluid ports is broken into its five constituent pieces and the electric port is broken into its three constituent parts.

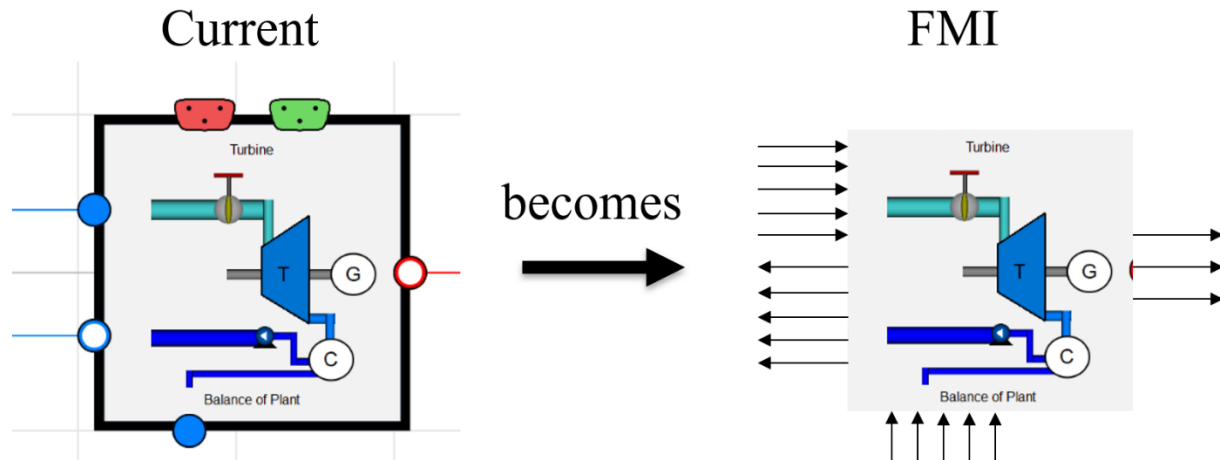


Figure 2. Transition from a Modelica physical model to a FMU.

In the HYBRID repository package structure, a set of adaptors have been created and added to the utility folder to allow a user to take an existing Modelica model and convert it into a model ready for export as a FMU. The package placement can be seen below in Figure 3. Further details on each FMI template and interface are outlined below.

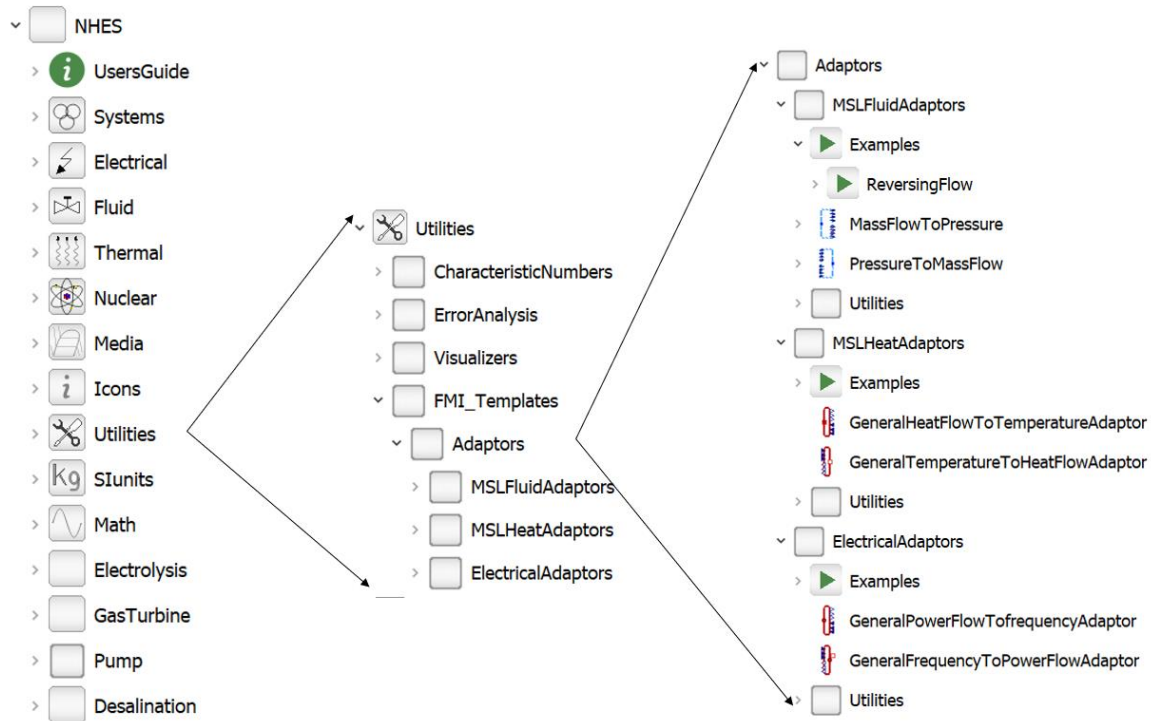


Figure 3. FMU Template folder location within the larger NHES folder.

### 2.1.1 Fluid Port Adaptors

Within the Utility.FMI\_Templates folder, there is an adaptor package created by specifically for Modelica standard library fluid adaptors, called MSLFluidAdaptors, that models acausal to causal adaptors. This folder was created in unison with Modelon as part of the FMI and FMU course's additional support hours. Within this folder, there are two adaptors, as shown in Figure 4. One of which is a pressure to mass flow adaptor, aptly named pressureToMassFlow. This adaptor's fluid port is best connected to a flow port of some sort (e.g., valves, resistance, pipe model). The inputs to this model are pressure at the interface, upstream enthalpy from the causal side, upstream mass fraction from the causal side, and upstream trace composition from the causal side. Outputs are acausal mass flow rate, upstream enthalpy from the acausal side, upstream mass fraction from the acausal side, and upstream trace composition from the acausal side.

The second adaptor is a mass flow to pressure adaptor, named MassFlowtoPressure adaptor. This adaptor's fluid port is best connected to a volume port (e.g., pressure sink, tank model). Inputs to this model are causal mass flow rate, upstream enthalpy from the causal side, upstream mass fraction from the causal side, and upstream trace composition from the causal side. Outputs are pressure at the interface, upstream enthalpy from the acausal side, upstream mass fraction from the acausal side, and upstream trace composition from the acausal side.

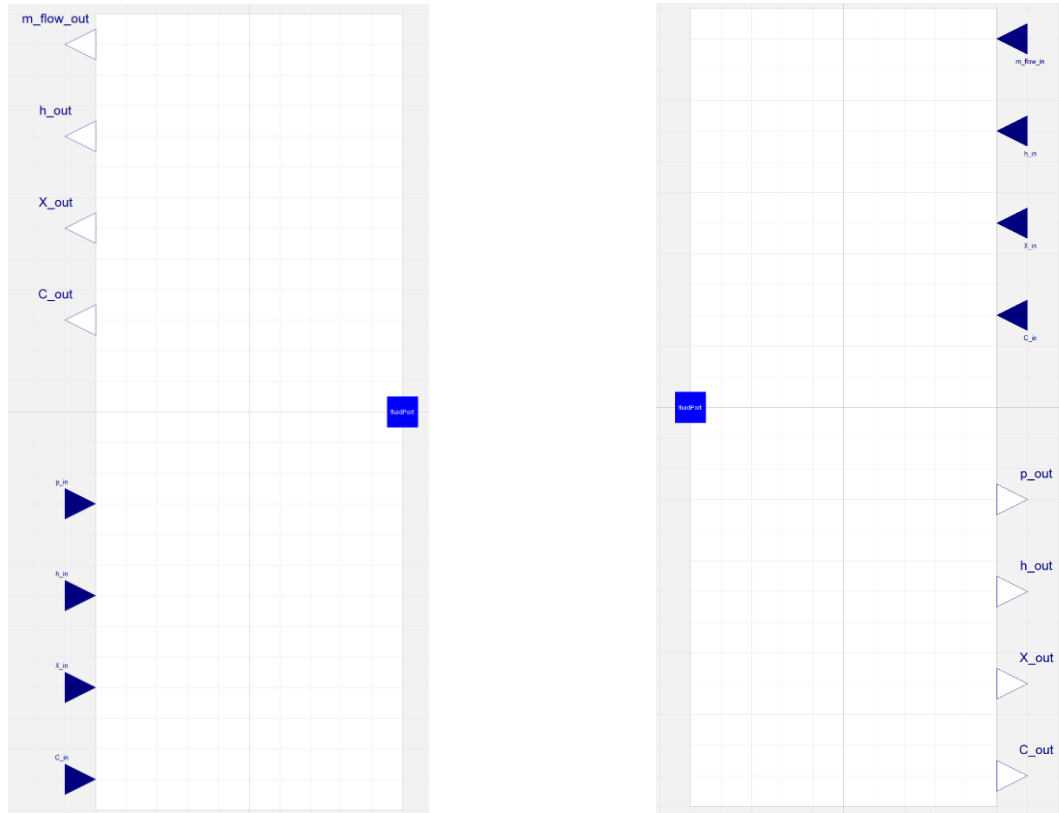


Figure 4. (Left) pressuretoMassFlow adaptor created by Modelon for use in the INL plug-and-play framework. This adaptor is best connected to a resistance port able to set the output mass flow rate. (Right) massFlowtoPressure adaptor created by Modelon for use in the INL plug-and-play framework. This adaptor is best connected to a volume port able to set the output pressure.

Figure 5 illustrates the usage of the two adaptors on a single model that involves reversible flow. The model is of a series of two fully open valves connected to a volume source between them and a pressure source on either side of the valves. The system fluid is moist air from the Modelica standard library. The pressure source is then subjected to a 1 Hz oscillatory frequency on the pressure system, as would be present in a fast-moving pressure chamber, while the pressure sink remains at a constant pressure. In normal operations, this system will have a reversible flow as the pressure of the source oscillates about the pressure sink pressure. Such a scenario has been a challenge to meet in the past with FMIs and FMUs due to the reversible nature of the mass flow. With the new adaptors, this reversible flow issue can be met.



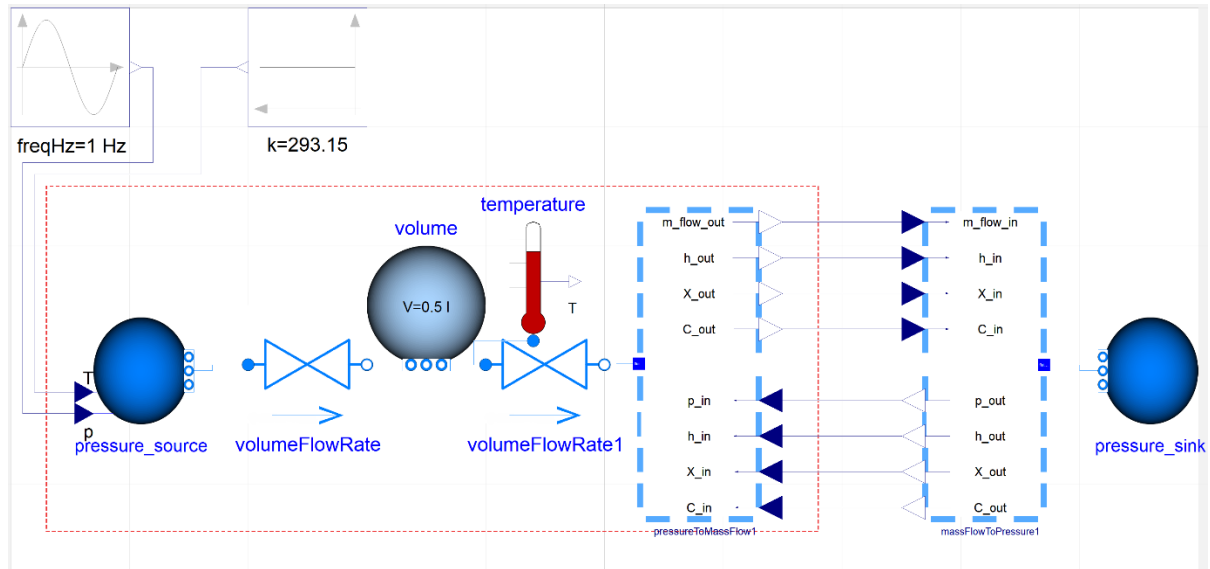


Figure 5. An example using adaptors that includes two pressure sources using moist air, one of which oscillates in pressure, causing a mass flow reversal. The red boxed in unit will become an FMU.

The red boxed area of Figure 5 has been exported as both a model exchange and cosimulation FMU, as shown in Figure 6 and Figure 7. All systems were then run for 10 seconds of simulation time. Results are depicted in Figure 8.

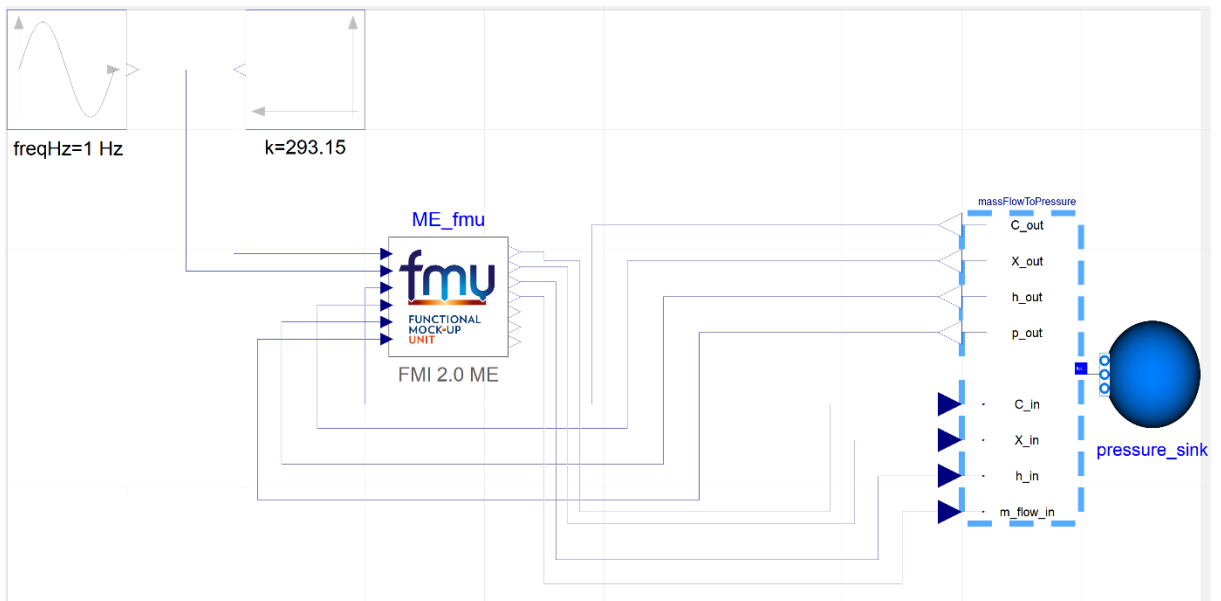


Figure 6. Example of a reversible flow using two pressures sources and moist air with a model exchange FMU.

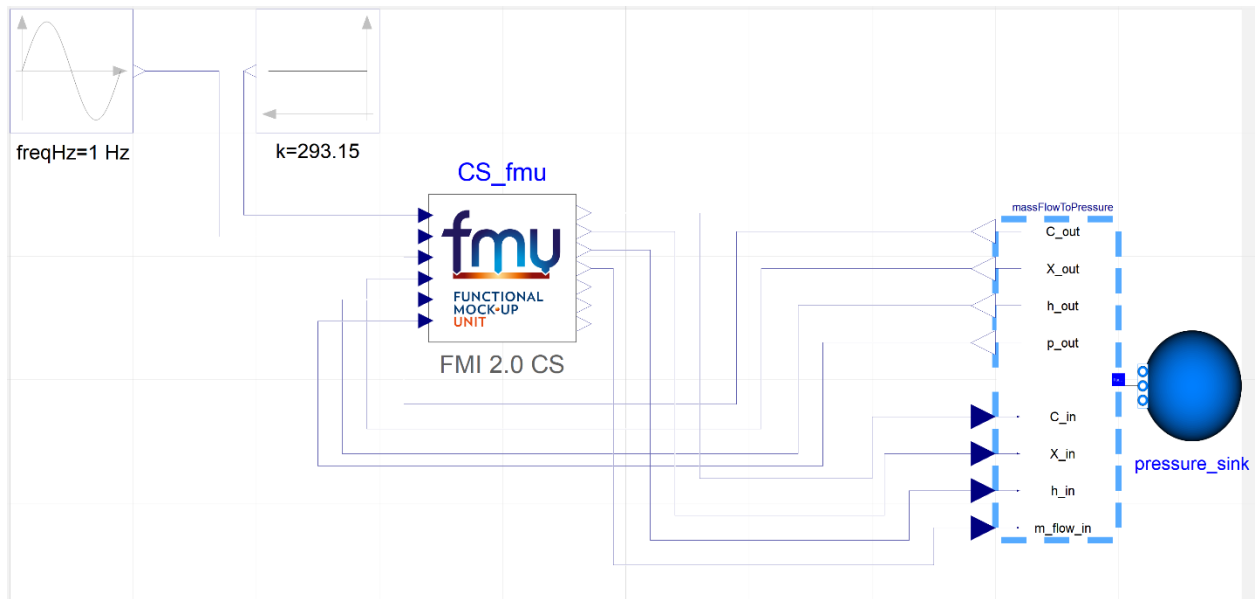


Figure 7. Example of a reversible flow using two pressures sources and moist air with a cosimulation FMU.

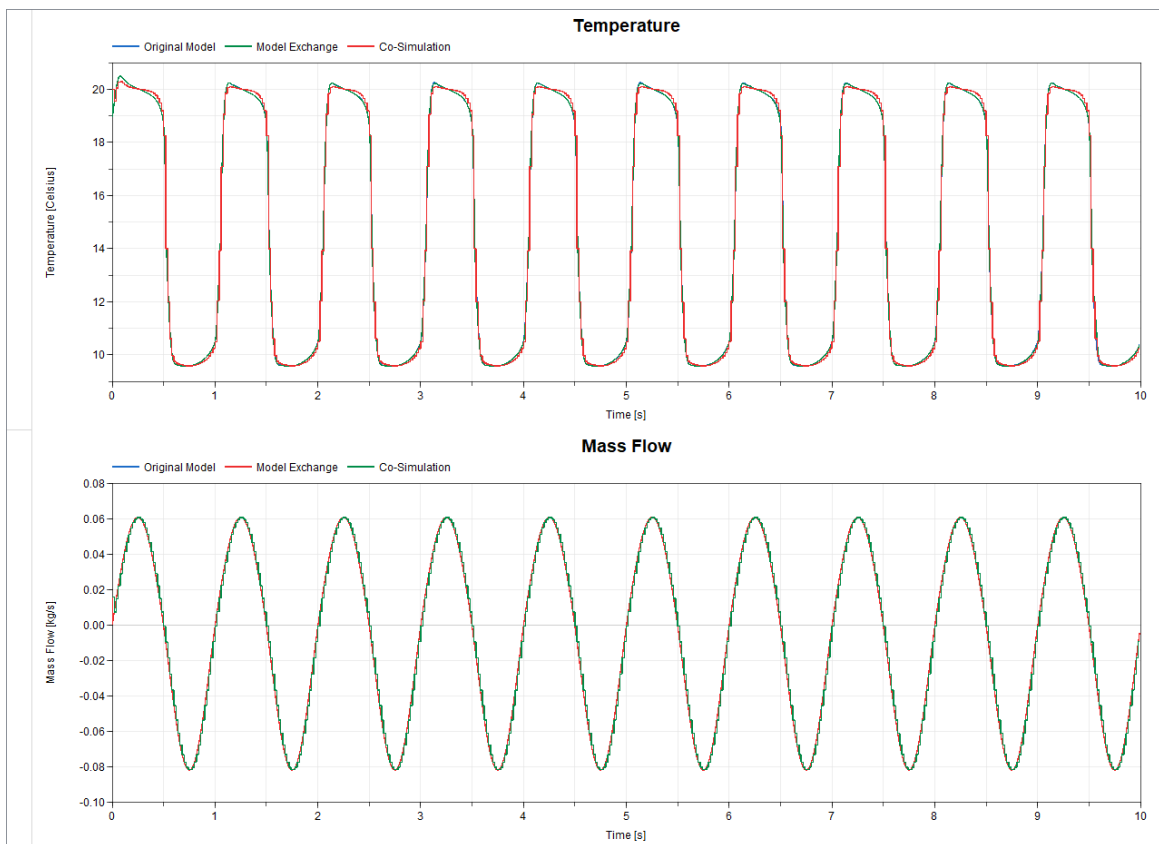


Figure 8. Mass flow to the pressure sink comparison across the original model, model exchange FMU, and cosimulation FMU (timestep = 0.02 seconds).

Results showcase that, by utilizing the fluid port adaptors, a reversible flow is achievable in FMIs and FMUs in both model exchange and cosimulation modes. However, these capabilities are not without an additional overhead for central processing unit (CPU) time. Model exchange for this particular model increases the simulation time from 2.364 seconds to 6.749 seconds. Cosimulation with a 0.02 second communication interval took 10.795 seconds. Even so, cosimulation still has the largest error. This is the limitation that occurs with cosimulation models because they are inherently an explicit solve. However, given a sufficiently small communication interval, given the dynamics of the model, an acceptable answer can be achieved.

### 2.1.2 Thermal Port Adaptors

Within the Utility.FMI\_Templates folder, there is an adaptor package created by specifically for Modelica standard library thermal adaptors, called MSLHeatAdaptors, that models acausal to causal adaptors. These models were initially made available in the Modelica standard library and have been augmented with additional examples and placed within the Nuclear Hybrid Energy Systems (NHES) package for ease of access relative to the other FMI adaptors. Two adaptors are included, one being the GeneralHeatFlowToTemperature adaptor. The inputs to this adaptor are the acausal heat flow port, causal heat flow, and optional causal first and second derivatives of heat flow. Outputs are temperature and the optional first and second derivative of temperature.

The second adaptor is the GeneralTemperaturetoHeatFlow adaptor. The inputs to this adaptor are the acausal heat flow port, causal temperature, and optional causal first and second derivatives of temperature. Outputs are heat flow and the optional first and second derivative of heat flow.

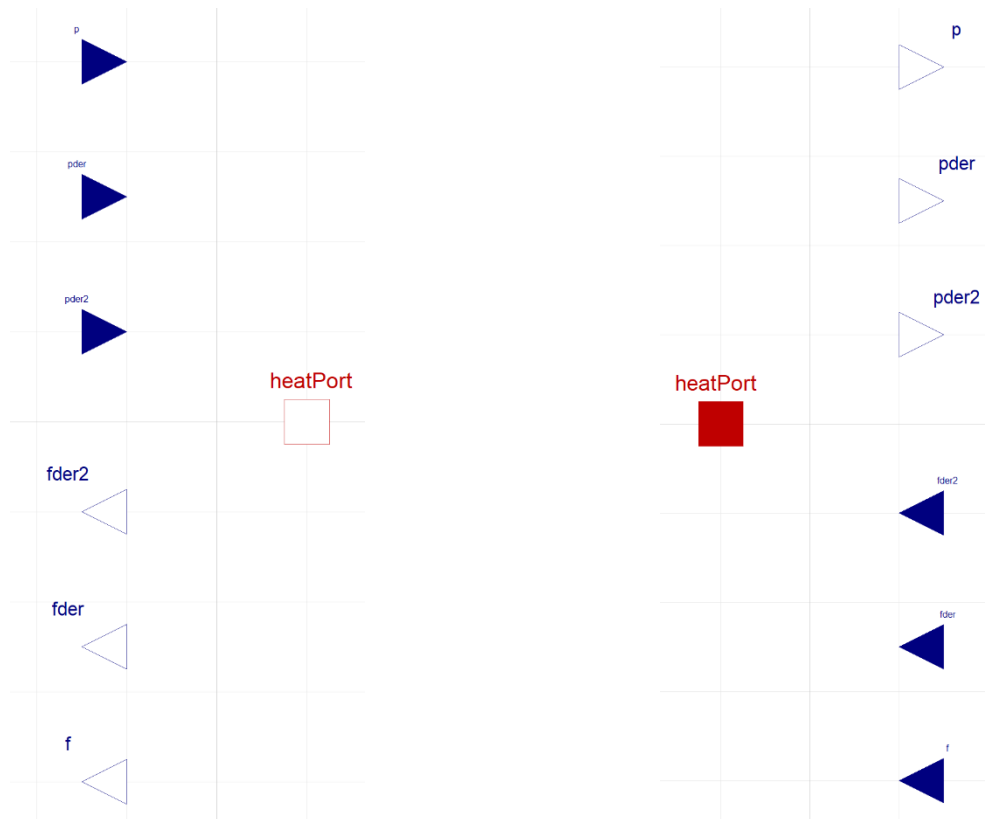


Figure 9. (Left) GeneralTemperatureToHeatFlow adaptor for use in the INL plug-and-play framework. (Right) GeneralHeatFlowToTemperature adaptor for use in the INL plug-and-play framework.

Figure 10 illustrates the usage of the two adaptors on a single model that involves two methods of using heat ports. The upper model demonstrates how to export two heat capacitors and connect them together in a target system. This requires that one of the capacitors (here: DirectCapacity) is defined to have states and that the temperature and derivative of temperature are provided in the interface. The other capacitor (here: InverseCapacity) requires a heat flow according to the provided input temperature and derivative of temperature. The lower part demonstrates how to export a conduction element that only needs temperatures for its conduction law and connects this conduction law in a target system between two capacitors. Both models will be translated into a model exchange and cosimulation model, as shown in Figure 11, Figure 12, Figure 13, and Figure 14, with results being compared in Figure 15 and Figure 16.

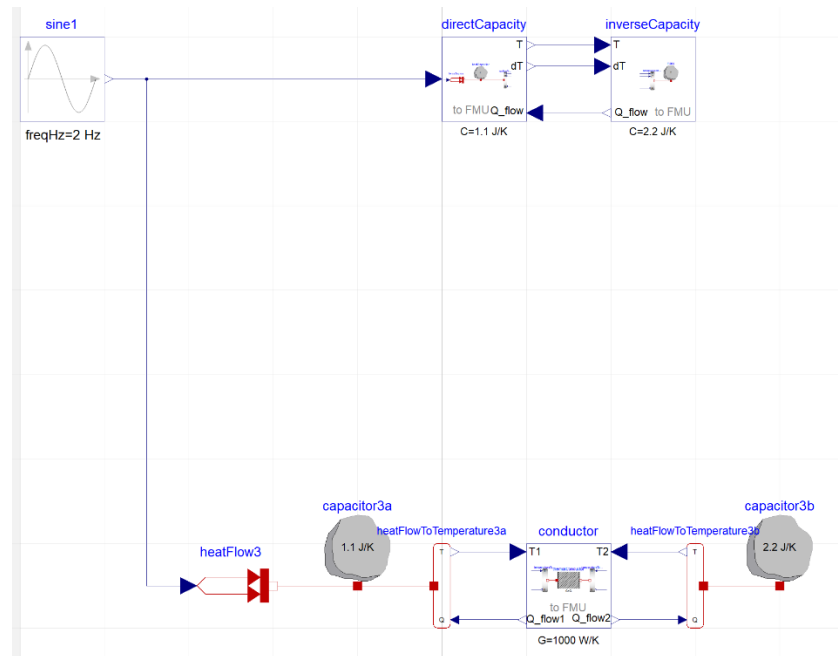


Figure 10. Example meant to demonstrate the FMU variants available with the thermal FMU adaptors. The upper part demonstrates how to export two heat capacitors and connect them together in a target system. The lower part demonstrates how to export a conduction element that only needs temperatures for its conduction law and connects this conduction law in a target system between two capacitors.

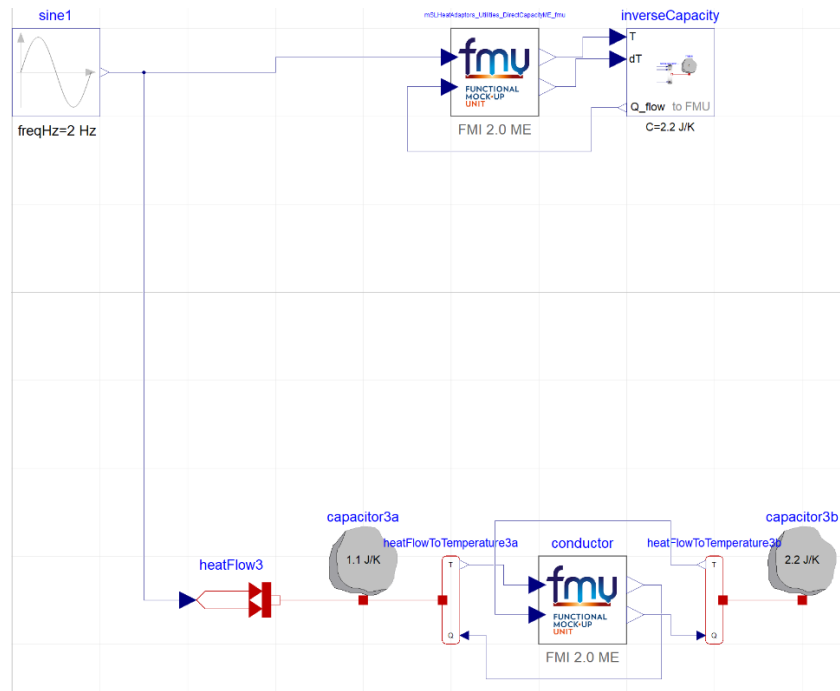


Figure 11. Demonstration of an FMU variant example using model exchange for thermal heat port adaptors.

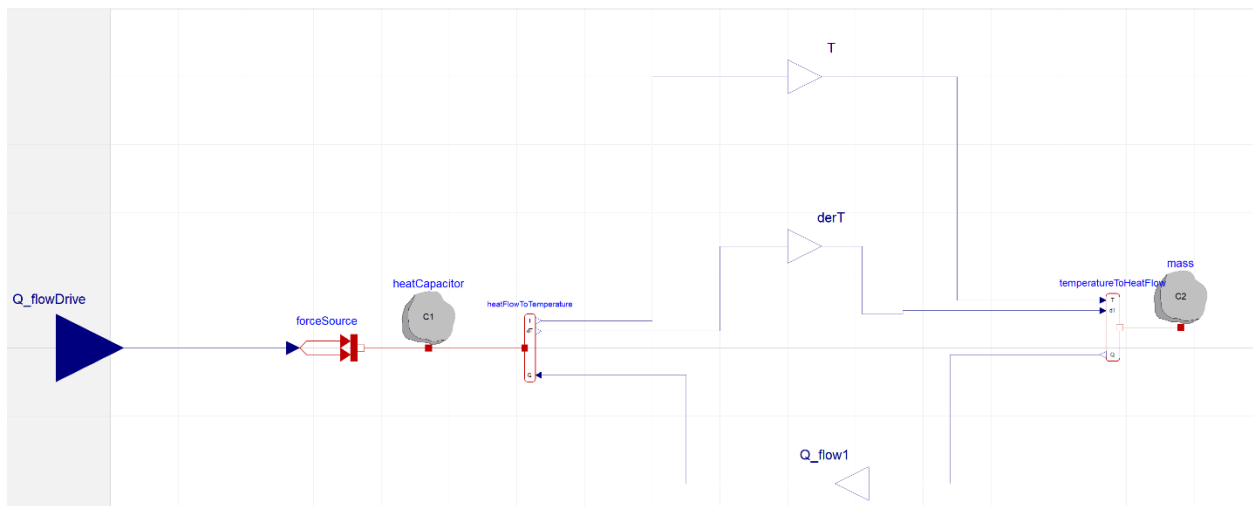


Figure 12. Collapse of the upper part into a single FMU for cosimulation. This is required because the frequency between the direct and inverse conduction problem is so fast that a single cut between the two could not be made without instabilities occurring.

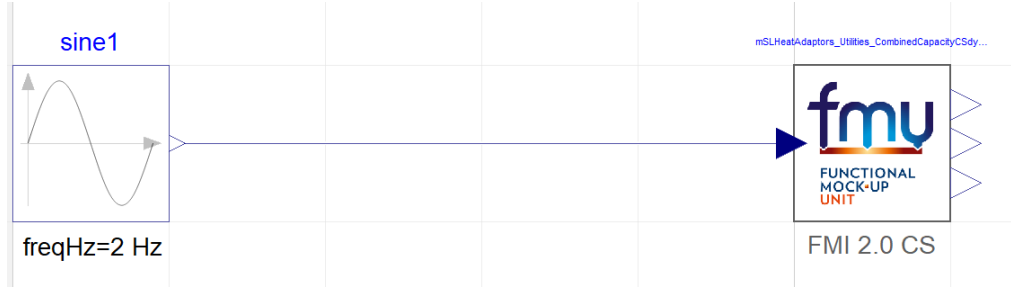


Figure 13. Upper model connected with the combined direct, inverse cosimulation FMU.

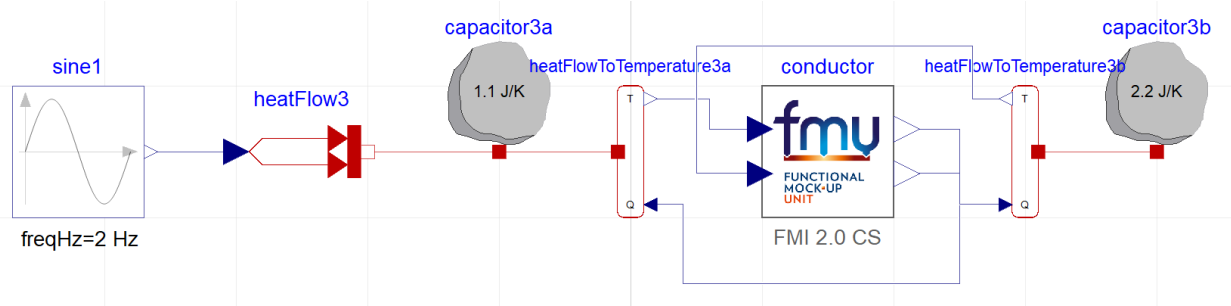


Figure 14. Lower model cosimulation FMU.

Results showcase that, by utilizing the thermal adaptors, heat flow between models with acceptable results is achievable in FMIs and FMUs in both model exchange and cosimulation modes. However, these capabilities are not without an additional overhead for CPU time, which is similar to the fluid port scenario. The cosimulation mode, while being the easiest to theoretically export to external codes due to its inclusion of a solver, required the most augmentation due to the fast system dynamics. This limitation required that the FMU be inclusive of both the direct and inverse capacitor within a singular model, as shown in Figure 12, otherwise a divergent solution was acquired. Even with this additional step, the cosimulation solve still has the largest error, as depicted in Figure 15 and Figure 16. This is because cosimulation models are inherently an explicit solve. However, given a sufficiently small communication interval and the dynamics of the model, an acceptable answer can be achieved.

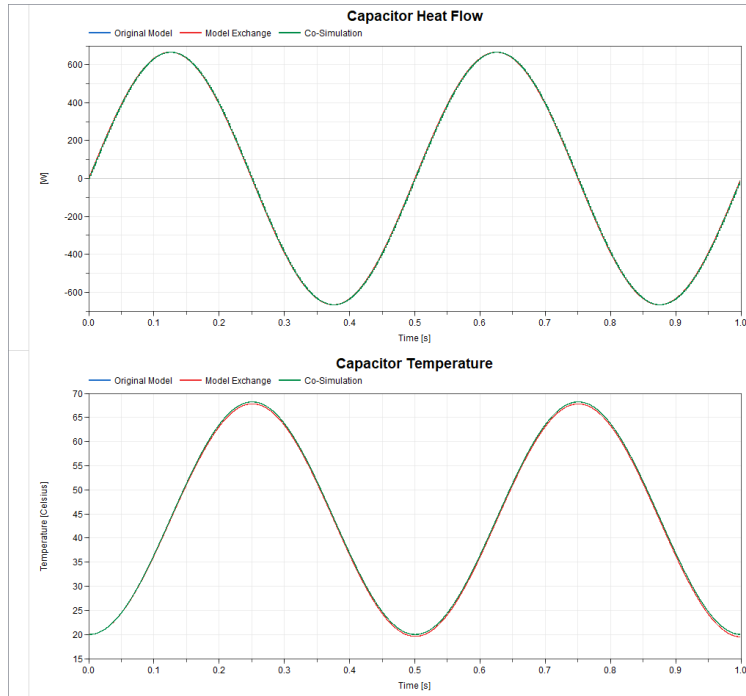


Figure 15. Direct/inverse simulation results for the original, model exchange, and cosimulation (communication interval 0.002 seconds) FMU runs. (Top) Heat flow into the capacitor. (Bottom) Capacitor 3b temperature.

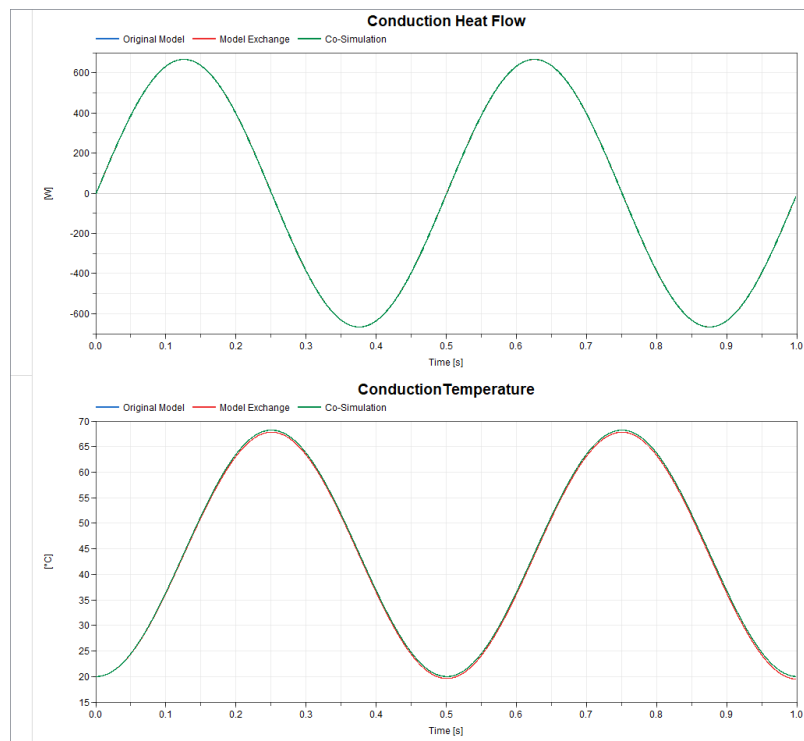


Figure 16. Conduction (lower model) simulation results for the original, model exchange, and cosimulation (communication interval 0.002 seconds) FMU runs. (Top) Heat flow into the capacitor. (Bottom) Capacitor 3b temperature.

### 2.1.3 Electrical Port Adaptors

Within the Utility.FMI\_Templates folder, there is an adaptor package created specifically for electrical adaptors, called ElectricalAdaptors, that models acausal to causal adaptors. Two adaptors are included, shown in Figure 17, one being the GeneralPowerFlowToFrequency adaptor. The inputs to this adaptor are the acausal electrical port, causal power, and optional causal first and second derivatives of power. Outputs are the frequency and the optional first and second derivative of frequency.

The second adaptor is the GeneralFrequencyToPowerFlow adaptor. The inputs to this adaptor are the acausal electrical port, causal frequency, and optional causal first and second derivatives of frequency. Outputs are the power flow and the optional first and second derivative of power flow.

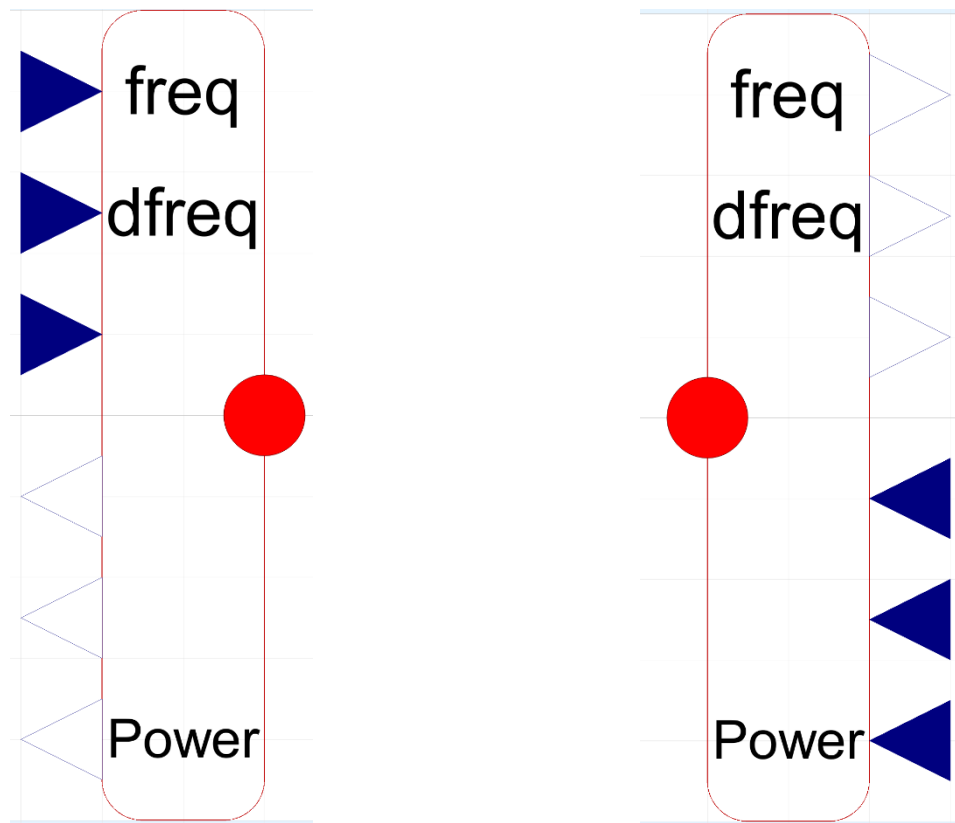


Figure 17. (Left) GeneralFrequencyToPowerFlow adaptor for use in the INL plug-and-play framework. (Right) GeneralPowerFlowTofrequency adaptor for use in the INL plug-and-play framework.

## 2.2 Functional Mock-Up Unit Options

In Dymola, there are several ways to export a model as an FMU, as shown in Figure 18. The FMU can include three different types of export: model exchange, cosimulation using the CCode solver, and cosimulation using various Dymola solvers. All the models can be exported as FMU binaries, while only model exchange and cosimulation using CCode can be exported as C-code due to proprietary information concerns. However, binaries are operating system dependent; therefore, care must be taken to ensure that the export of binary FMUs is on the same operating system.



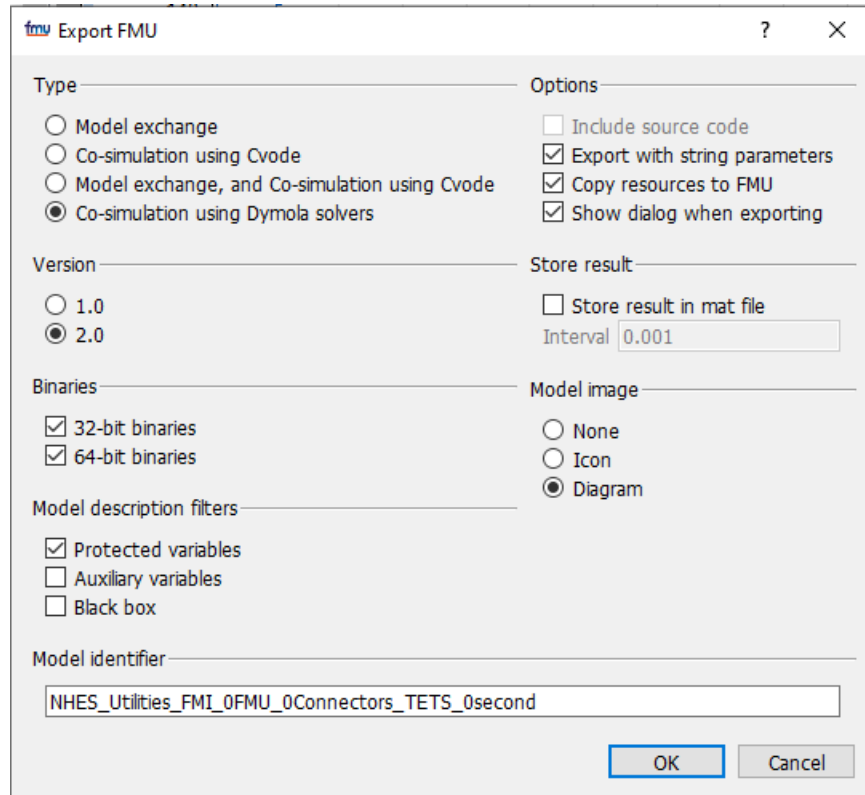


Figure 18. FMI and FMU export options from Dymola2020FD01.

## 2.3 Full-Scale IES Model

Through the creation of FMUs, it becomes possible to encapsulate a model from one coding language in a standardized format usable within another coding language. To test this functionality, a NuScale-style reactor set was chosen. The modeling set, shown in Figure 20, includes the reactor, energy manifold, turbine generator, and electric grid all modeled in the Modelica language. The turbine generator set was then converted from a Modelica model into an FMU to ensure that all the proper data is input into and transferred between the models. The initial step being to implement the adaptors mentioned in the previous section that transforms the fluid ports into constituent real outputs, as shown in Figure 19. The progression of translation is shown in Figure 20, going from the Modelica only model to a model exchange FMU that is then included in the model.

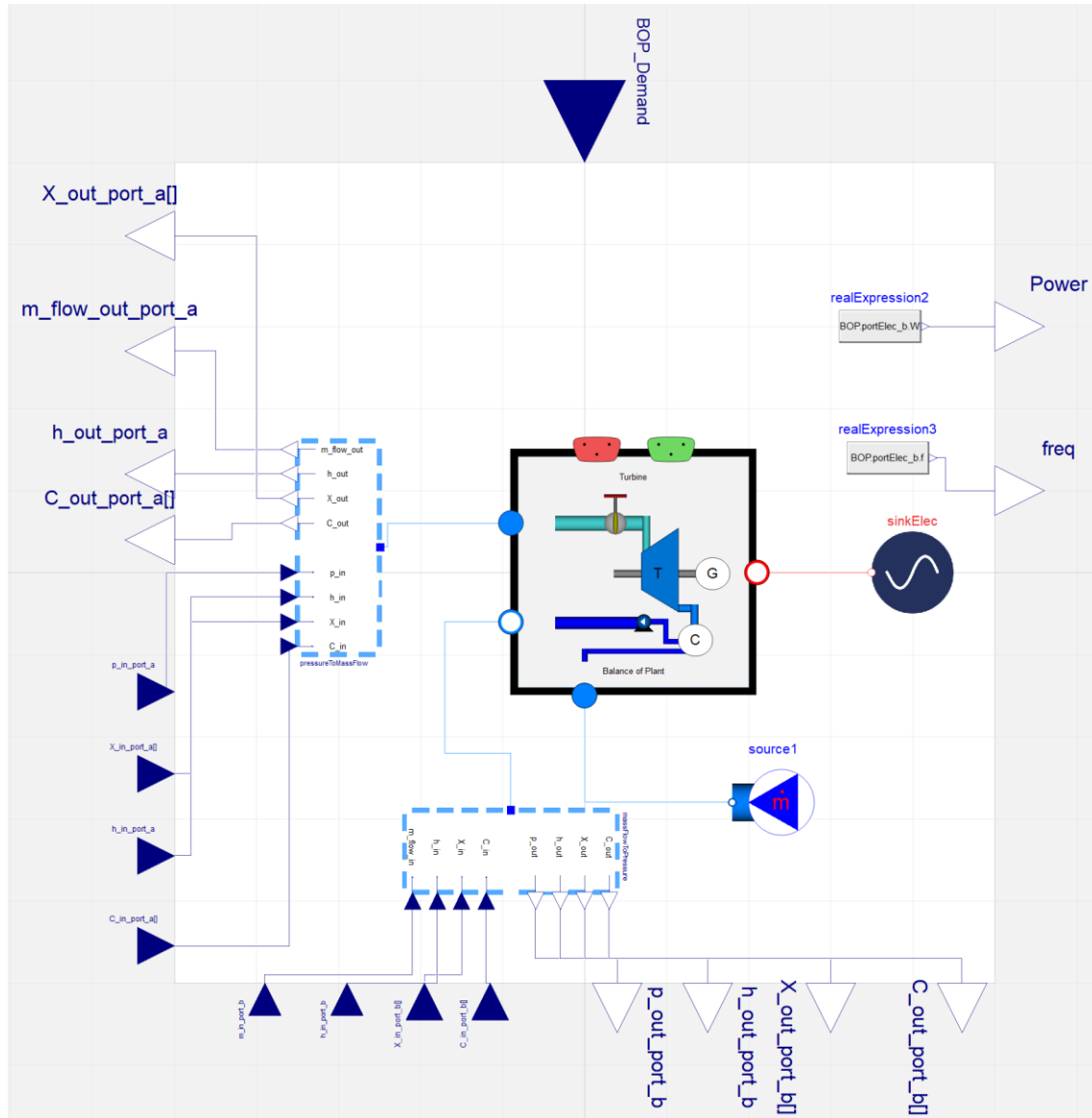


Figure 19. Translation of Modelica Turbine Generator Model into an FMU ready design.

The control system within the turbine generator model is maintained through the translation process and can fulfill the desired setpoints within the turbine model. Then, the model is exported into a model exchange FMU and reimported into the Modelica framework. A comparison of the turbine power output is depicted in Figure 21, showing that the different versions of the model are closely matched. Differences can be attributed to minute differences in initialization subroutines that occur in the initialization phase of the run. FMU-based results and input-based Modelica results are nearly identical, and both simulations were able to meet turbine demand setpoints. It is worth noting that a version using cosimulation was attempted, but instabilities from the explicit time-stepping scheme could not be overcome, and cosimulation was ultimately deemed unsolvable. Such scenarios become more common as the complexity of the models increases. While cosimulation is the easiest version of FMI to implement, instabilities such as these also increase the possibility of simulation roadblocks.

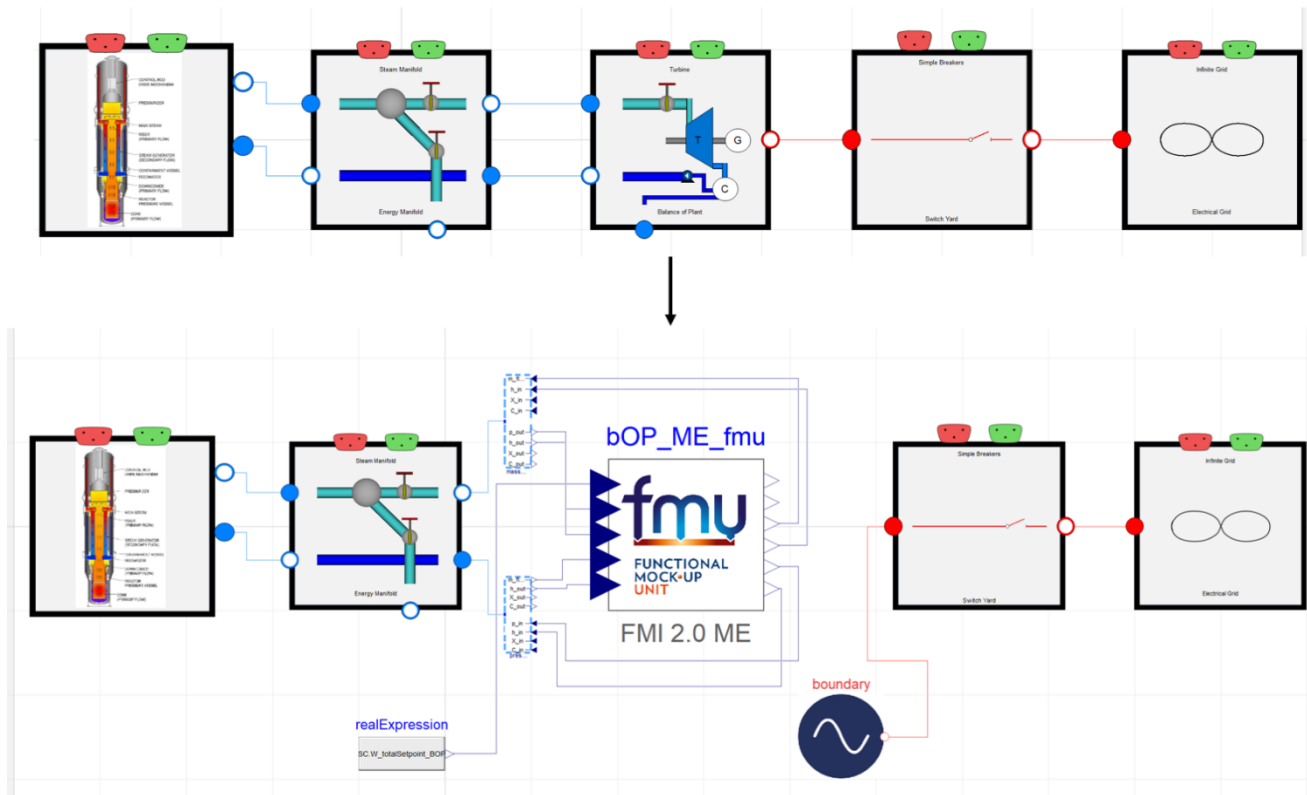


Figure 20. Transition from a Modelica model to an input-based Modelica model to an FMI-based simulation.

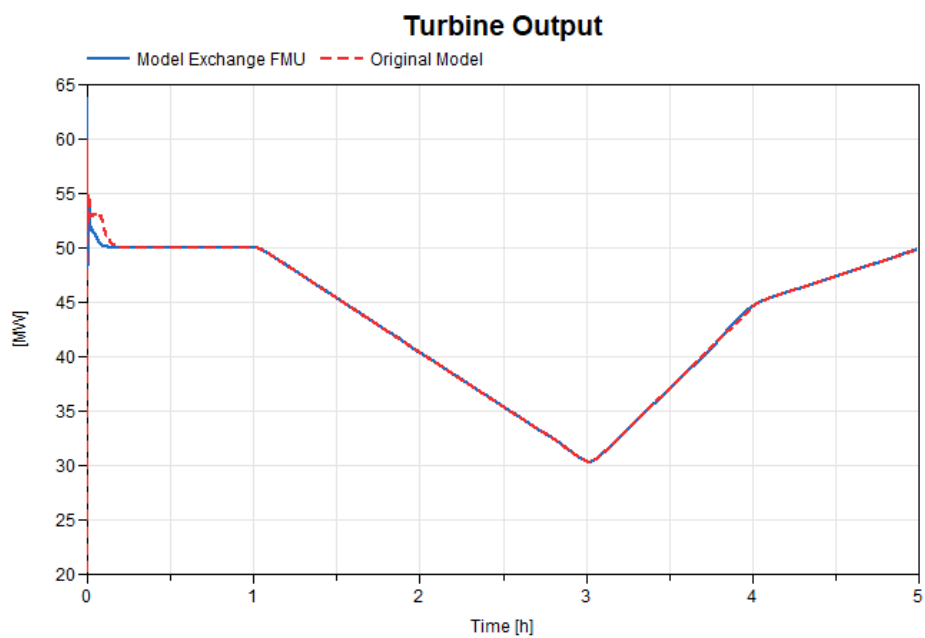


Figure 21. Comparison of turbine output results between the original model and model exchange FMU.

### 3. HYBRID REPOSITORY

At the beginning of the IES project, a version control repository was delivered in order to have a common location for the deployment of system and component models and analyses, developed and constructed with Modelica/Dymola and RAVEN. In order to initiate the activity of the construction of a flexible plug-and-play Modelica/RAVEN framework for IES analysis, a restructuring of the version control repository (namely HYBRID, available at <https://hpcgitlab.inl.gov/hybrid/hybrid>) has been performed.

The following main tasks have been performed for this specific activity:

- Usage of the RAVEN regression test system (named ROOK) for the deployment of a single, integrated testing platform for both Modelica and Dymola models and analysis and RAVEN workflows. The testing system has been linked with the automatic continuous integration tool for the automatic testing of the models and analyses when new modifications are added in the repository.
- Optimization of the folder structures for easier browsing and usage of the version control repository.

Figure 22 shows the new repository structure, where the following main folders can be identified:

- **Models**: containing the Modelica and Dymola models
- **TRANSFORM library**: Submodule of the Oak Ridge based TRANSFORM library that is used as the base models for many of the integrated energy systems models
- **raven**: linking to the RAVEN repository
- **scripts**: containing the *dymola\_launcher* and the tools for loading Modelica and Dymola outputs into a Python environment
- **tests**: containing all the tests that are automatically executed by the continuous integration system and executable, locally, running the command “run\_tests.”














Name	Last commit	Last update
 .gitlab	issue and MR templates added	2 years ago
 Models	Removing more ThermoPower Requirements in the Two-Tank ...	1 month ago
 archive	added description	1 month ago
 developer_tools	added description	1 month ago
 scripts	removed typo	23 hours ago
 tests	no mat	2 weeks ago
 TRANSFORM-Library @ 96d3493a	added transform submodule	1 month ago
 raven @ 02e4a71e	submodule	1 month ago
 .gitignore	removed obsolete steam manifold models... updated to new ...	3 years ago
 .gitmodules	added transform submodule	1 month ago
 00README.txt	Civet test dirctory set-up	4 years ago
 README.md	Update README.md	4 months ago
 run_tests	Use proper pathsep for the os.	1 month ago

Figure 22. New structure of the repository.

Additionally, a series of Modelica tests have been added to test the system-level interactions in the NHES Modelica repository. An example output of the regression system is shown in Figure 23.

Table 1. Synopsis of Modelica test cases.

<b>Test</b>	<b>Description</b>
<b>GTTP_Test</b>	Gas Turbine Load Follow Test – 60 second electric demand oscillation
<b>NSSS_test</b>	Westinghouse 4-Loop Test – 10,000 seconds at nominal power
<b>NuScale_4Loop</b>	Test of load following a NuScale reactor – 5-hour load follow simulation
<b>Simple_Breakers_Test</b>	Test of electrical breakers on an infinite grid
<b>Test_Battery_Storage</b>	Test of a simple electrical battery system – logical power flow simulation
<b>Test_Thermal_Storage</b>	Test of a Therminol-66 thermal energy storage facility through both charge and discharge cycles
<b>Desalination 1 Pass</b>	Single Stage Reverse Osmosis Component Check
<b>Desalination 2 Pass</b>	Second State Reverse Osmosis Component Check
<b>Desalination 2 Pass Mixing</b>	Two Stage Reverse Osmosis with mixing
<b>Desalination Reverse Osmosis Module</b>	Fully Encapsulated Two Stage Reverse Osmosis with mixing
<b>Desalination NHES Basic</b>	Controlled Desalination NHES system
<b>Desalination NHES Complex</b>	Controlled with signal bus NHES RO system with parallel osmosis units
<b>NuScale Primary Test</b>	Testing of the Primary loop of the NuScale loop
<b>NuScale Nominal Test</b>	Addition of Nominal Power test for the NuScale reactor
<b>HTSE Power Test</b>	HTSE NHES system based on power input control
<b>HTSE Steam Test</b>	HTSE NHES system based on steam and power input control
<b>Generic Modular PWR</b>	Small modular reactor of a NuScale size system with a pump
<b>BOP Boundaries Test A</b>	Balance of plant system based on pressure difference
<b>BOP Boundaries Test B</b>	Balance of plant system based on forced mass flow rate
<b>Step-Down Turbines</b>	Basic set of step-down turbines
<b>Step-Down Turbines Complex</b>	Test of a more complex step-down turbine system
<b>TightlyCoupled_FY18_Battery</b>	Complex system of systems from the 2018 case (including Electric Battery Storage)
<b>Tightly Coupled_FY18_TES</b>	Complex system of systems from the 2018 case (including thermal energy storage [two-tank sensible heat])
<b>Supervisory Control Test</b>	Test of the Supervisory control system for input from external files

While these tests are not exhaustive of the Modelica repository system, they provide a systems-level understanding of the repository model state. Additional tests will be added on an “as-needed” basis.

```

    diffMatrices = np.abs(varTrajectories - varTrajectoriesgold)
rook: Loading init file "C:/msys64/home/FRICKL/cleaning_hybrid/hybrid/scripts/rook.ini"
rook: ... loaded setting "add_non_default_run_types = dymola,raven"
rook: ... loaded setting "add_run_types = dymola,raven"
rook: ... loaded setting "test_dir = tests"
rook: ... loaded setting "testers_dirs = scripts/testers,raven/scripts/TestHarness/testers/"
rook: found 27 test dirs under "tests" ...
rook: Loading init file "C:/msys64/home/FRICKL/cleaning_hybrid/hybrid/scripts/rook.ini"
rook: ... loaded setting "add_non_default_run_types = dymola,raven"
rook: ... loaded setting "add_run_types = dymola,raven"
rook: ... loaded setting "test_dir = tests"
rook: ... loaded setting "testers_dirs = scripts/testers,raven/scripts/TestHarness/testers/"
(1/27) Success( 35.33sec)tests\dymola_tests\BOP_L1_Boundaries_a_Test\
(2/27) Success( 35.97sec)tests\dymola_tests\BOP_L1_Boundaries_b_Test\
(3/27) Success( 13.83sec)tests\dymola_tests\Desalination_1_pass\
(4/27) Success( 15.08sec)tests\dymola_tests\Desalination_2pass_mixing\
(5/27) Success( 13.84sec)tests\dymola_tests\Desalination_2_pass\
(6/27) Success( 22.37sec)tests\dymola_tests\Desalination_NHES_basic\
(7/27) Success( 19.87sec)tests\dymola_tests\Desalination_R0module\
(8/27) Success( 37.94sec)tests\dymola_tests\Desalination_NHES_complex\
(9/27) Success( 15.19sec)tests\dymola_tests\GTP_Test\
(10/27) Success( 32.14sec)tests\dymola_tests\Generic_Modular_PWR\
(11/27) Success( 21.18sec)tests\dymola_tests\HTSE_Power_Test\
(12/27) Success( 28.29sec)tests\dymola_tests\HTSE_Steam_Test\
(13/27) Success( 34.55sec)tests\dymola_tests\NSSS_test\
(14/27) Success( 46.37sec)tests\dymola_tests\NuScale_4Loop\
(15/27) Success( 29.77sec)tests\dymola_tests\NuScale_Nominal_Test\
(16/27) Success( 13.34sec)tests\dymola_tests\Simple_Breakers_Test\
(17/27) Success( 23.47sec)tests\dymola_tests\NuScale_primary_test\
(18/27) Success( 13.91sec)tests\dymola_tests\StepDownTurbines\
(19/27) Success( 15.16sec)tests\dymola_tests\StepDownTurbines_complex\
(20/27) Success( 11.99sec)tests\dymola_tests\Supervisory_Control_Test\
(21/27) Success( 12.02sec)tests\dymola_tests\Test_Battery_Storage\
(22/27) Success( 29.38sec)tests\dymola_tests\Test_Thermal_Storage\
(23/27) Success( 31.79sec)tests\dymola_tests\Thermocline_Cycling\
"failing"
(24/27) Skipped( None! )tests\dymola_tests\TightlyCoupled_FY18_Battery\
"failing"
(25/27) Skipped( None! )tests\dymola_tests\TightlyCoupled_FY18_TES\
(26/27) Success( 22.59sec)tests\dymola_tests\Thermocline_Insulation\
(27/27) Success( 28.13sec)tests\raven_tests\train\TrainArmaOnData

PASSED: 25
SKIPPED: 2
FAILED: 0
(raven_libraries)

```

Figure 23. An example of tests run in the ROOK regression system.

In addition to adding tests to the regression system, additional capabilities have been added to allow for a smoother cross-platform and cross-machine compatibility. These capabilities were necessary because the commercial platform Dymola by Dassault systems has a series of flags that cause different outputs to be sent to the final solution file. Ensuring every user has the same flags turned on and off is unrealistic because some of the flags are global settings turned on for every simulation loaded into their particular instantiation of Dymola. To get around this, the ROOK testing system has added the capability to only look at the time steps or time intervals that are guaranteed to be included in each simulation of the model, regardless of the flags automatically loaded by Dymola. To accomplish this, an extra option, either “*numberOfIntervals*” or “*OutputInterval*,” is required in the `simulateModel` command in the regression system. The option `numberOfIntervals` tells Dymola how many output intervals to make, whereas `OutputInterval` tells Dymola at what timestep interval an output should be present for comparison. These can be selected in the Simulation Setup tab of the Dymola GUI.

Also, a restart file loading capability has been added to the Modelica regression system. This has been included because, for complex models, the initialization phase of a simulation can take the Modelica solvers a significant amount of time to find an initialization point. This occurs due to the highly nonlinear nature of the underlying physical equations. A way to avoid such situations is to provide a restart file to bypass the initialization phase of the simulation. A restart file is automatically created at the end of each simulation, as the `dsfin.txt` file created in the folder where the simulation is run. This file includes the final values of the previous simulation from which the new model can restart. Moving this file to the tests gold folder and loading it into the regression system can save substantial time in regression testing and provide a consistent starting point for each test rather than relying on the same initialization point to be

found during each regression testing cycle. Full details on how to utilize and create new regression tests can be found on the hybrid wiki at (<https://hpcgitlab.inl.gov/hybrid/hybrid>).

The work previously reported is propaedeutic for the release of the modeling framework in the open-source community. Several activities have been deployed for the upcoming open-sourcing of the software:

- User documentation:
  - Development of an extensive user manual, covering the detailed description of the models (Modelica and Dymola) and instructions on how to execute them
- Software quality assurance (SQA) documentation (see Figure 24). Such documentation is aimed to collect the following information:
  - Project planning information
  - High-level overview touching on our entire project and software development
  - Roles and responsibilities
  - Merge request workflow (code change requests)
  - Workflow diagram
  - Plan for developing the software
  - Document references to other relevant plans and procedures
  - Contains information about the software safety determination and quality level determination
  - Definitions of software V&V
  - Methods and procedures for the software V&V

And it is composed by the following set of documents:

- HYBRID Software Quality Assurance Plan, which details the SQA procedures adopted for the development and life cycle of the HYBRID software framework
- HYBRID Software Configuration Management Plan
- HYBRID Software Test Plan
- HYBRID IT Asset Maintenance Plan
- HYBRID Verification and Validation Plan
- HYBRID Software Design Description
- HYBRID Software Requirement Specification
- HYBRID Traceability Matrix
- HYBRID Configuration Item List

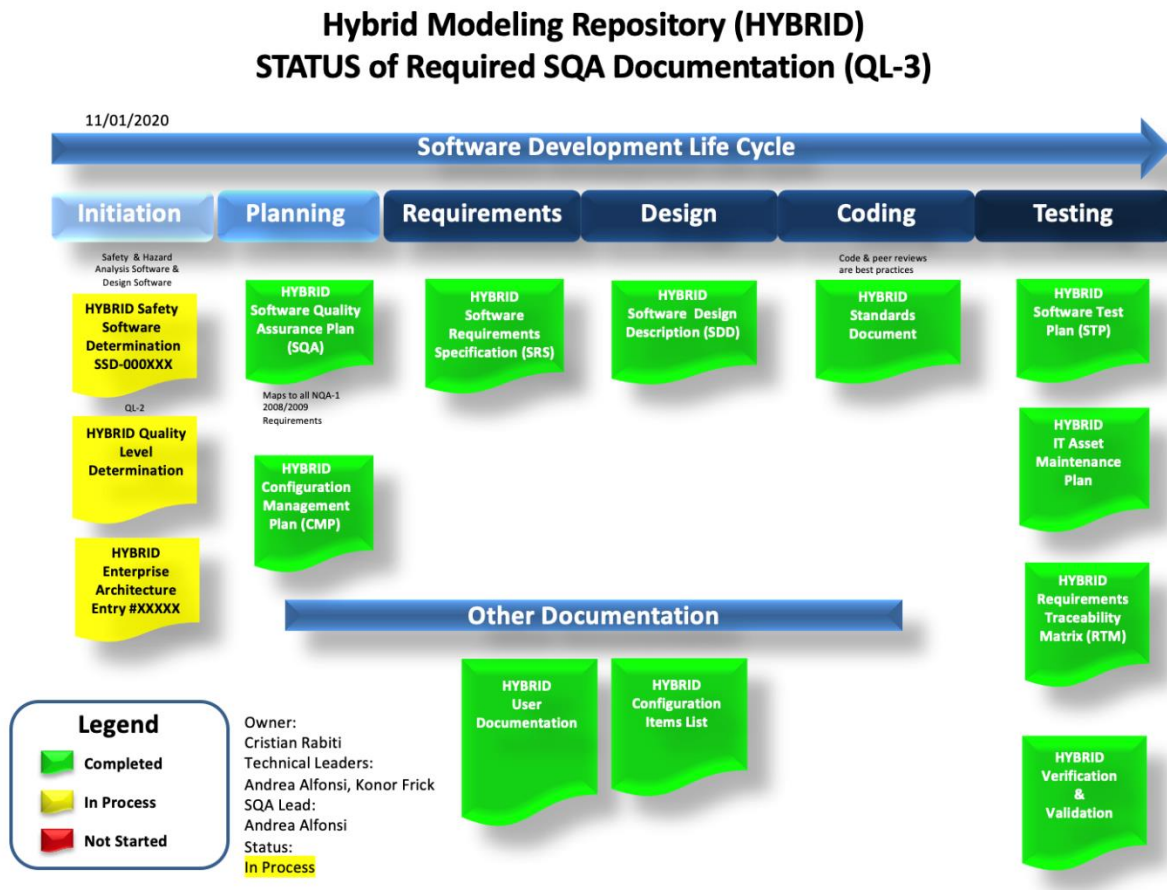


Figure 24. Status of the required SQA documentation for the HYBRID modeling repository.

## 4. IMPLEMENTATION IN RAVEN

In previous milestone reports [4],[5], it has been demonstrated the successful execution of the FMIs and FMUs using external Python-based framework (FMpy [7] and PyFMI [8]). Such showcase provided the basis for the implementation of the FMI and FMU interfaces within the RAVEN framework.

In RAVEN, the coupling with system and physical models is performed by the model entity application program interface (API). The model entity represents a “connection pipeline” between the input and the output space. The RAVEN framework provides APIs for the following main model categories:

- Codes
- Externals
- Reduced order models
- Hybrid models
- Ensemble models
- Postprocessors

For the deployment of the FMI and FMU system, the externals (external models) is key.



The external model (see Figure 25) allows the developer to create, in a Python module or platform, a direct coupling with, for example, a model coded in Python (e.g., a set of equations representing a physical model, connection to another code, control logic). Once the external model is constructed, it is interpreted and used by RAVEN and, at run-time, becomes part of RAVEN itself.

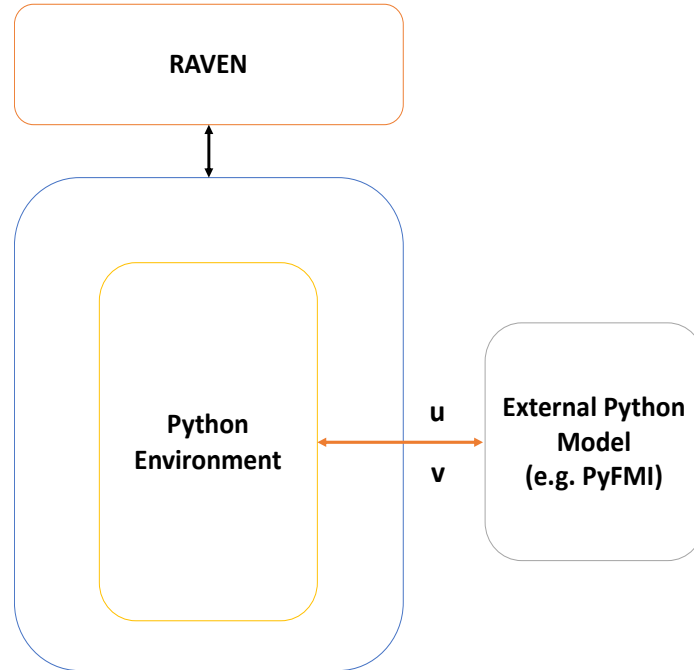


Figure 25. External model API.

Depending on the type of protocol for the FMI or FMU of interest, two coupling schemes with the RAVEN code are under development.

Figure 26 shows the coupling scheme under development when the FMI or FMU cosimulation protocol needs to be used; RAVEN interacts with the different models via PyFMI or FMpy (or similar importer package). In this coupling scheme, RAVEN “perceives” the models imported via FMIs/FMUs as any other external model or code.

On the other end, Figure 27 shows the coupling scheme under development when the FMI/FMU model exchange protocol is used; in this configuration, RAVEN is able to directly interact with the universal solver that is aimed at solving all the models (imported via FMI/FMUs). This coupling scheme is the preferable one overall for the interaction with artificial intelligence (AI) algorithms.

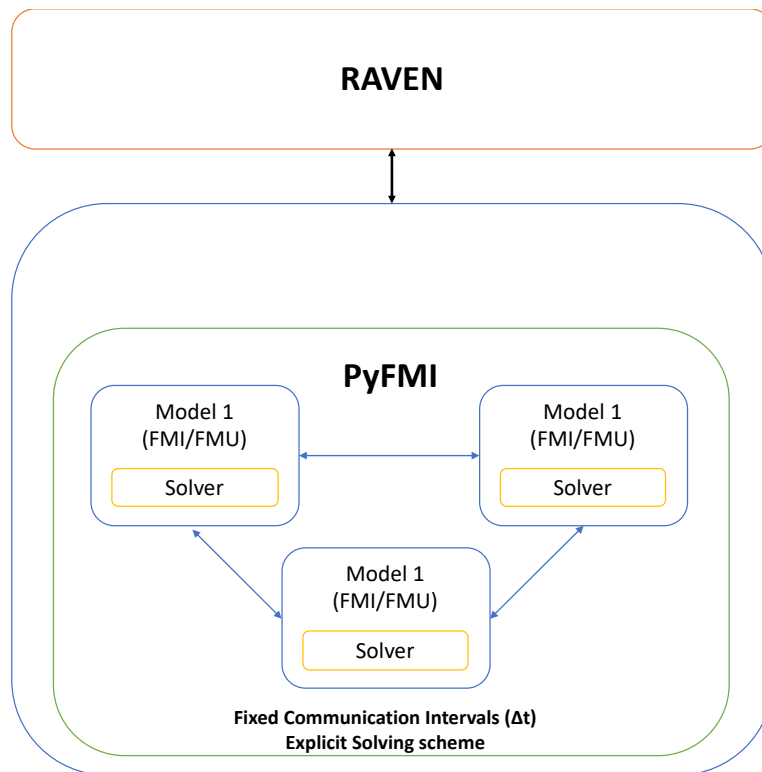


Figure 26. FMI/FMU cosimulation protocol coupled with RAVEN.

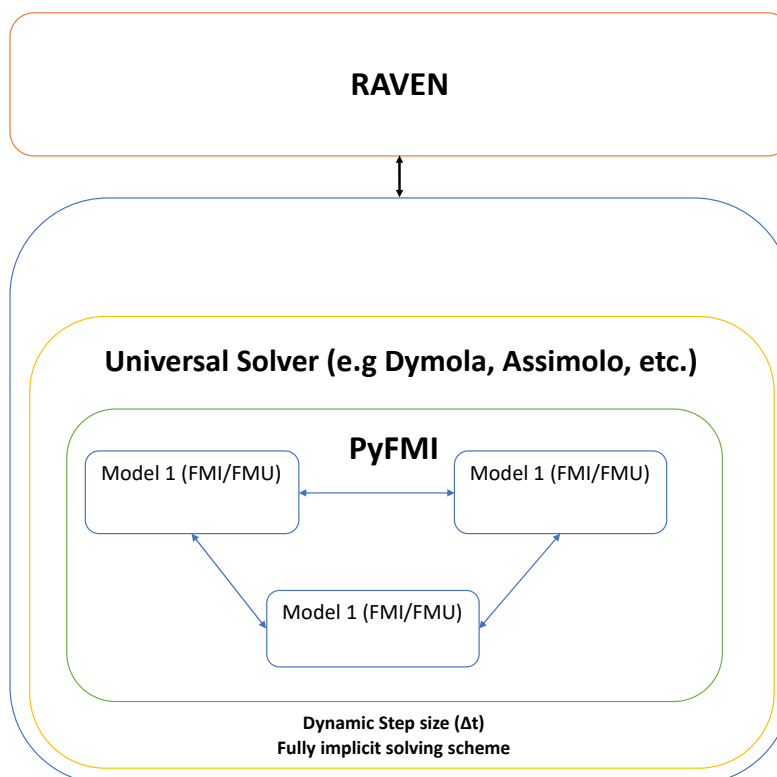


Figure 27. FMI/FMU model exchange protocol coupled with RAVEN.

## 5. ACCELERATION OF FMI/FMU MODELS VIA ARTIFICIAL INTELLIGENCE

As mentioned in the previous section, the RAVEN framework provides APIs for different model categories, among which is the hybrid model. The hybrid model is a special class of model that is aimed to couple in-tandem, high-fidelity physical and mathematical models (e.g., FMI/FMU Dymola models) and artificial intelligence algorithms (e.g., surrogate models). The AIs are trained based on the results from the high-fidelity model. The global accuracy of the AIs is evaluated based on cross-validation scores, and the local (e.g., prediction) validity is determined via some local validation metrics (e.g., crowding distance). Once the AIs are trained, the hybrid model can decide which of the models (i.e., the AIs or high-fidelity model) to execute based on the previously mentioned accuracy and validation metrics. Figure 28 shows the scheme behind the hybrid model formulation.

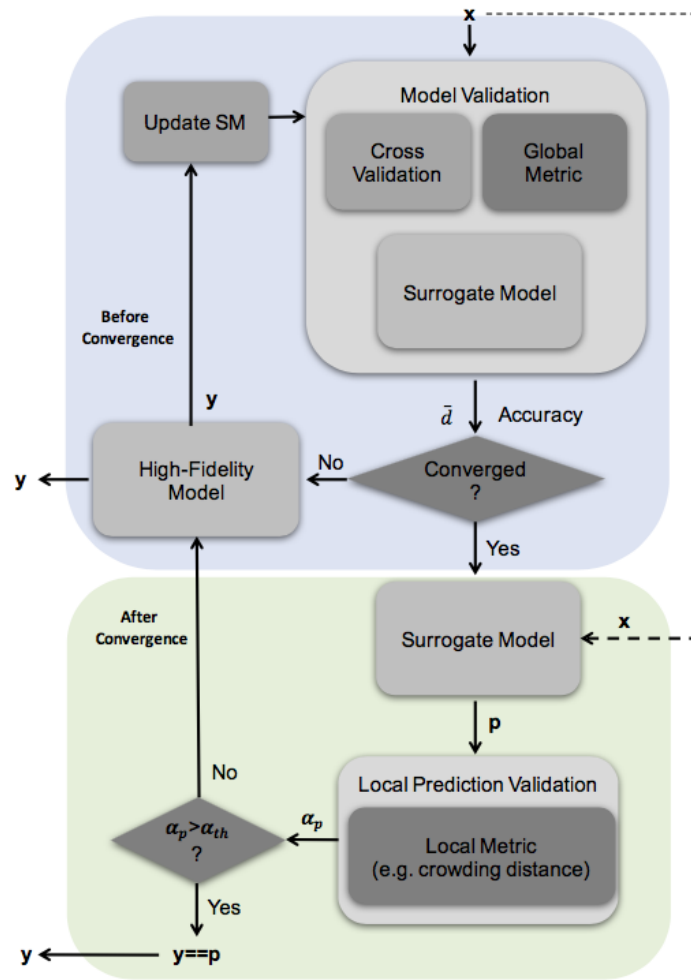


Figure 28. RAVEN hybrid model scheme.

For this project, we proposed leveraging such a capability in order to accelerate the execution of systems composed by several FMI/FMUs. Each of these FMI/FMUs will be “coupled” in a hybrid model configuration, resulting in a potential CPU time saving.

## 6. CONCLUSION

This report discussed the status of the flexible plug-and-play framework development currently ongoing that aims to integrate Modelica and Dymola with RAVEN in terms of both FMI/FMU construction and repository structures that aim to ease the sharing and simulation of complex dynamic models.

The report provides an in-depth look at the required model alterations needed to modify existing system-level models to be exported as an FMU. These alterations include modifying specialty “port” variables into their constituent parts as real variables via a new FMI adaptor package that has been added to the existing HYBRID repository. This package includes new adaptors for electrical, fluid, and heat ports for export into the FMI/FMUs. Examples have been included within the FMU adaptor package illustrating how to properly utilize the system. Several of these examples were discussed within Section 2 of this report.

Simulation results demonstrate that, while minor differences may occur, the overall control, trends, and solution integrity is maintained between standard Modelica simulation and FMU simulation results. However, it is worth noting that, for small systems, the FMU results have a slower simulation time than the Modelica only simulation. While this step-by-step process does require several steps of checks, it does provide a level of system flexibility that has not been present before. Using this process, a company can provide models that contain proprietary information to separate entities without disclosing any of the information about the model that could be considered business sensitive. Such an ability would allow institutions to bypass the necessity of “whitewashing” data.

In addition to the investigative work being conducted on FMUs and FMIs, a series of updates to the hybrid repository regression system has been completed to get the repository ready for open-sourcing. These updates include additional system-level tests for components in the hybrid repository. Increasing the testing level from a mere six tests to 25 and counting. Further, features to the testing system have been included, such as an initialization subroutine for Dymola models that helps highly nonlinear complex systems begin their regression test. Additionally, output keys “numberOfIntervals” and “OutputInterval” have been added to the regression system that allow for consistent comparison points between the gold file and the simulation results between machines. This is necessary because the commercial Modelica platform Dymola has a series of global output flags that are utilized and are rarely consistent between organizations, yet they do not change the trajectories of the solution.

Overall, extensive work has been completed on developing FMUs and FMIs from existing models and on understanding the requirements and limitations of FMI/FMUs. Additional investigative work is planned to expand the FMU capabilities within the existing hybrid repository framework.

## 7. REFERENCES

- [1] Dassault Systems. “DYMOLA Systems Engineering: Multi-Engineering Modeling and Simulation Based on Modelica and FMI.” Accessed July 24, 2020. <https://www.3ds.com/products-services/catia/products/dymola/>.
- [2] Alfonsi, A., C. Rabiti, D. Mandelli, J. Cogliati, C. Wang, P. W. Talbot, D. P. Maljovec, and C. Smith. 2016. “RAVEN Theory Manual and User Guide.” INL/EXT-16-38178, Idaho National Laboratory.
- [3] Rabiti, C., A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita, S. Sen, C. Wang, and J. Chen. 2017. “RAVEN User Manual.” INL/EXT-15-34123, Idaho National Laboratory.
- [4] Alfonsi, A., K. Frick, S. Greenwood, and C. Rabiti. 2020. “Status on the Development of the Infrastructure for a Flexible Modelica/RAVEN Framework for IES.” INL/EXT-20-00160, Rev 00, Idaho National Laboratory.
- [5] Frick, K., A. Alfonsi, C. Rabiti. 2020. “Flexible Modelica/RAVEN Framework for IES.” INL/EXT-20-00419, Rev 00, Idaho National Laboratory.
- [6] Modelica. 2010. “Functional Mockup Interface (FMI).” January 2010.
- [7] FMpy. 2020. “FMpy 0.2.21.” Accessed July 8, 2020. <https://pypi.org/project/FMPy/>.
- [8] PyFMI. 2018. “PyFMI 2.5.” Accessed March 25, 2020. <https://pypi.org/project/PyFMI/>.