ORNL/SPR-2015/17

Update on Small Modular Reactors Dynamic System Modeling Tool–Web Application



Approved for public release.
Distribution is unlimited.

Richard E. Hale
M. Sacit Cetiner
David L. Fugate
John J. Batteh,
Modelon, Inc.
Michael M. Tiller,
Xogeny Corporation

January 2015

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express 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. Reference 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 United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/SPR-2015/17

Reactor and Nuclear Systems Division

UPDATE ON SMALL MODULAR REACTORS DYNAMIC SYSTEM MODELING TOOL–WEB APPLICATION

Richard E. Hale
M. Sacit Cetiner
David L. Fugate
John J. Batteh, Modelon, Inc.
Michael M. Tiller, Xogeny Corporation

Date Published: January 2015

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

Page

LIST OF FIGURES v
ACRONYMS vii
ACKNOWLEDGMENTS ix
EXECUTIVE SUMMARY xi
1. INTRODUCTION 1
1.1 BACKGROUND 1
2. WEB-BASED APPLICATION 3
2.1 INTRODUCTION/TASK DESCRIPTION 3
2.2 WEB-BASED APPLICATION DESCRIPTION 3
2.2.1 Web-Based Application Deployment/Hosting 5
2.2.2 Web-based Application Authentication and Authorization 5
2.2.3 Web-based Application Structure/File Needs 5
2.2.4 Simplified Web-Based Application Updating 6
2.2.5 Integration with Central Modeling Repository 7
2.2.6 Simplified User Interface 7
2.3 WEB-BASED APPLICATION MODELS 7
2.3.1 Liquid Metal Reactor Models 7
2.3.2 Fluoride High-Temperature Reactor Models—Update 9
2.3.3 Intermediate Heat Transport System 13
2.4 INSTRUMENTATION AND CONTROLS OVERLAYS AND SIMULATION 14
2.4.1 Fluoride High-Temperature Reactor Example Transient Simulations 16
2.5 WEB-BASED APPLICATION USE 16
2.6 FUTURE WEB-BASED APPLICATION DEVELOPMENT 21
2.6.1 Online Simulation 21
2.6.2 FMI Development 24
3. MODEL COLLABORATION 27
3.1 INTRODUCTION/TASK DESCRIPTION 27
3.2 COLLABORATION PLATFORM 27
3.2.1 ORNL-ARM 27
3.3 COLLABORATION MODELS 28
3.3.1 DRACS/RVACS System Models 28
3.3.2 Primary Heat Transport System Models 28
3.3.3 Intermediate Heat Transport System Models 28
3.3.4 Heat Exchanger Components Models 29
3.3.5 Steam Generator Component Models 29
3.3.6 Power Conversion System Models 29
3.3.7 Grid Models 29
3.3.8 Instrument and Controls and Event Driver Models 30
3.4 COLLABORATION WORKFLOW 30
3.4.1 Example Subsystem Model Test Rig (Electrical) 31
REFERENCES 35
APPENDIX A. CREATING/UPDATING WEB-BASED APPLICATION A-1

LIST OF FIGURES

Figure **Page**

Fig. 1. Modular architecture for advanced liquid metal reactor end-to-end system model. 4
Fig. 2. Modular architecture for fluoride high-temperature reactor end-to-end system model..... 4
Fig. 3. PRISM system design. 8
Fig. 4. Modular architecture for Advanced Liquid Metal Reactor end-to-end system model. 8
Fig. 5. Small Modular Advanced High-Temperature Reactor system design. 9
Fig. 6. Modular architecture for fluoride high-temperature reactor end-to-end system model..... 9
Fig. 7. Current top-level integrated end-to-end system model of the Small Modular Advanced High-Temperature Reactor plant. 10
Fig. 8. Single-zone model of the reactor, fuel, and the connected coolant system. 11
Fig. 9. Dual-zone model of the reactor, fuel, and the connected coolant system..... 11
Fig. 10. Modelica diagram layer of the Small Modular Advanced High-Temperature Reactor intermediate heat exchanger model. 12
Fig. 11. Temperature profiles on the shell and tube sides of the Small Modular Advanced High-Temperature Reactor intermediate heat exchanger. 13
Fig. 12. Modelica diagram layer of the Small Modular Advanced High-Temperature Reactor intermediate heat transport system. 14
Fig. 13. Instrumentation and controls conceptual diagram. 14
Fig. 14. Instrumentation and controls Modelica diagram. 15
Fig. 15. Control loops for primary heat transport system reactivity control and primary cooling pump. 16
Fig. 16. Reactivity disturbance example simulation. 17
Fig. 17. Intermediate heat transport system flow disturbance example simulation. 17
Fig. 18. Web-based application reactor architecture selection. 18
Fig. 19. Web-based application subsystem architecture selection. 18
Fig. 20. Web-based application subsystem parameter selection. 19
Fig. 21. Web-based application Excel download (FMI add-in). 19
Fig. 22. Web-based application Excel option (FMI add-in). 20
Fig. 23. Excel experiment sheet (FMI add-in). 20
Fig. 24. Automatic Excel plot generation (FMI add-in). 21
Fig. 25. iPad display for subsystem architecture selection. 22
Fig. 26. iPhone display for subsystem architecture selection. 23
Fig. 27. iPad display for subsystem parameter selection. 23
Fig. 28. iPhone display for subsystem parameter selection. 24
Fig. 29. Details for event driver module. 30
Fig. 30. Example test rig development package. 31
Fig. 31. Electrical component generator model test rig. 32
Fig. 32. Dymola dialogue for duplicating model. 32
Fig. 33. Dymola implementation of new model with clutch..... 33
Fig. 34. Dymola dialogue for extending model class..... 33
Fig. 35. Dymola model inserted into test rig..... 34
Fig. 36. Signal addition to model test rig. 34

ACRONYMS

ALMR	advanced liquid-metal reactor
ART	Advanced Reactor Technology
DOE	US Department of Energy
DoE	Design of Experiment
DRACS	direct reactor auxiliary cooling system
ED	event driver
FHR	fluoride high-temperature reactor
FMI	Functional Mock-up Interface
FMIE	Functional Mock-up Interface for Excel
FMU	Functional Mock-up Unit
I&C	instrumentation and controls
IHTS	intermediate heat transport system
IHX	intermediate heat exchanger
ORNL	Oak Ridge National Laboratory
PHTS	primary heat transport system
PRISM	power reactor innovative small module
QA	quality assurance
SISO	single-input–single-output
RVACS	reactor vessel auxiliary cooling system
SmAHTR	small modular advanced high-temperature reactor
SMR	small modular reactor
VBA	Visual Basic for Application

ACKNOWLEDGMENTS

The research described in this report was sponsored by the Advanced Reactor Technology (formerly the Advanced Reactor Concept) Program of the US Department of Energy Office of Nuclear Energy.

EXECUTIVE SUMMARY

The Small Modular Reactor (SMR) Dynamic System Modeling Tool project is in the third year of development. The project is designed to support collaborative modeling and study of various advanced SMR (non-light water cooled) concepts, including the use of coupled reactors at a single site.

In FY2015, the project has transitioned from the Advanced Small Modular Reactors Research and Development Program to the Advanced Reactors Technology Program. The objective of the project; however, remains the same; namely to provide a common simulation environment and baseline modeling resources to facilitate rapid development of dynamic advanced reactor models, ensure consistency among research products within the Instrumentation, Controls, and Human-Machine Interface technical area, and leverage cross-cutting capabilities while minimizing duplication of effort.

The combined simulation environment and suite of models are identified as the Modular Dynamic SIMulation tool. The critical elements of this effort include (1) defining a standardized, common simulation environment that can be applied throughout the program; (2) developing a library of baseline component modules that can be assembled into full plant models using existing geometry and thermal-hydraulic data; (3) defining modeling conventions for interconnecting component models; and (4) establishing user interfaces and support tools to facilitate simulation development (i.e., configuration and parameterization), execution, and results display and capture.

Previous deliverables focused on the development of component and system models as well as end-to-end system models using Modelica and Dymola for two advanced reactor architectures: (1) Advanced Liquid Metal Reactor and (2) fluoride high-temperature reactor (FHR). The focus of this deliverable is the release of the first beta version of the web-based application for model use and collaboration, as well as an update on the FHR model. The web-based application allows novice users to configure end-to-end system models from preconfigured choices to investigate the instrumentation and controls implications of these designs and allows for the collaborative development of individual component models that can be benchmarked against test systems for potential inclusion in the model library. A description of this application is provided along with examples of its use and a listing and discussion of all the models that currently exist in the library.

The remaining work in FY2015 includes the addition of preliminary models for advanced gas cooled reactor designs as well as the development of models that include additional I&C control for an end-to-end plant concept.

1. INTRODUCTION

1.1 BACKGROUND

As documented previously (Refs. 1–3), the intent of this project is to develop simulation resources and tools to allow collaborative modeling and control study for various advanced (non-light water reactor), small modular reactor (SMR) configurations. The project has been funded under the Advanced SMR Program and is in the third and final year of development. This program, however, is currently being transitioned to the Advanced Reactor Technology (ART) Program with less emphasis on SMRs and more emphasis on advanced reactor concepts. As a result, any further development will be transitioned to the ART Program priorities.

High-level objectives of this effort include (1) defining a standardized, common simulation environment that can be applied throughout the program; (2) developing a library of baseline component modules that can be assembled into full plant models; (3) defining modeling conventions for interconnecting component models (i.e., a standard architecture for simulation development); and (4) establishing user interfaces and support tools to facilitate simulation development (i.e., configuration and parameterization), execution, and results display and recording. Objectives 1–3 have been met and are documented for two different advanced reactor architectures in previous updates (Refs. 1–3). The user interfaces for novice, intermediate, and expert users have been developed. The purpose of this report is to document the integration of these interfaces into a web-based application for potential users.

2. WEB-BASED APPLICATION

2.1 INTRODUCTION/TASK DESCRIPTION

Simulations with Dymola and the Functional Mock-up Interface (FMI) tool demonstrate the flexibility of the modeling approach and architecture to study the behavior of different feedback measurements, control strategies, and transient events. However, both of these methods require the user to have access to and licenses for Dymola and/or Modelon's FMI Add-In to Excel. A goal of the project is to remove as many restrictions as possible from the use of advanced reactor models. To this end, the creation of a web-based application that allows permitted users to access models and share licenses transparently will greatly simplify the process and provide for greater collaboration among designers and developers across a broad spectrum of research programs both within and external to the US Department of Energy (DOE).

Previously, a prototype of a web-based application was developed for internal use and development [2]. The initial prototype was used to identify the necessary tools, files, and protocols that would be needed for a web-based application that would be distributed for wider use. The principal focus of this report is the description of and instructions for use for this web-based tool. Although intended for use by all, the target audience for the web-based application is the beginner user with little to no familiarity with Modelica/system models. More detailed analysis would be expected for using the FMI tool separately (by the more advanced intermediate user) and directly in Dymola (for the advanced user). A detailed discussion of the activities associated with intermediate and advanced users can be found in Ref. 2. A discussion of the use of the web-based application for the beginning user follows.

The beginning user is assumed to have no experience or proficiency in modeling activities and is considered to be an end user who is interested in running simple scoping analyses on existing models. He or she is not presumed to have access, familiarity with, or licenses for Modelica solvers such as Dymola. Additionally, the beginning user is expected to either work from pre-generated Functional Mock-up Interface for Excel (FMIE) simulation runs or to have access to shareable licenses for the FMIE tool. These scoping analyses allow the user to modify basic parameters and a small subset of advanced parameters and generate a predefined or reduced set of results through a web-based interface. The intention of this interface is to provide a robust set of features that do not allow the inexperienced user to cause failure or hang up the tool. This ensures a positive experience for the user and is intended to encourage continued use and development of more advanced modeling, such as that of novice and advanced users.

2.2 WEB-BASED APPLICATION DESCRIPTION

The development of a web-based application has been made possible in large part through the use of a modular architecture approach to end-to-end advanced reactor system models, as shown in Figs.1–2. The adoption of this approach follows those established in other Modelica-based system integration applications, such as the automotive industry. The development of individual component models that fit into a structured architecture in a plug-and-play format allows a user to make drop down selections which can be coded in a web-based application. The early prototype for the web-based application required preconfigured models with Functional Mock-up Units (FMUs) that could be selected and pulled into the FMI Add-In to Excel tool developed by Modelon, Inc. The major advancements in this beta version of the web-based application include the following upgrades, which are described in the subsections below:

1. multiple platform application deployment/hosting,
2. user authentication and access control,
3. simplified web-based application file structure/needs,
4. simplified web-based application updating,
5. integration with central repository, and
6. simplified user interface.

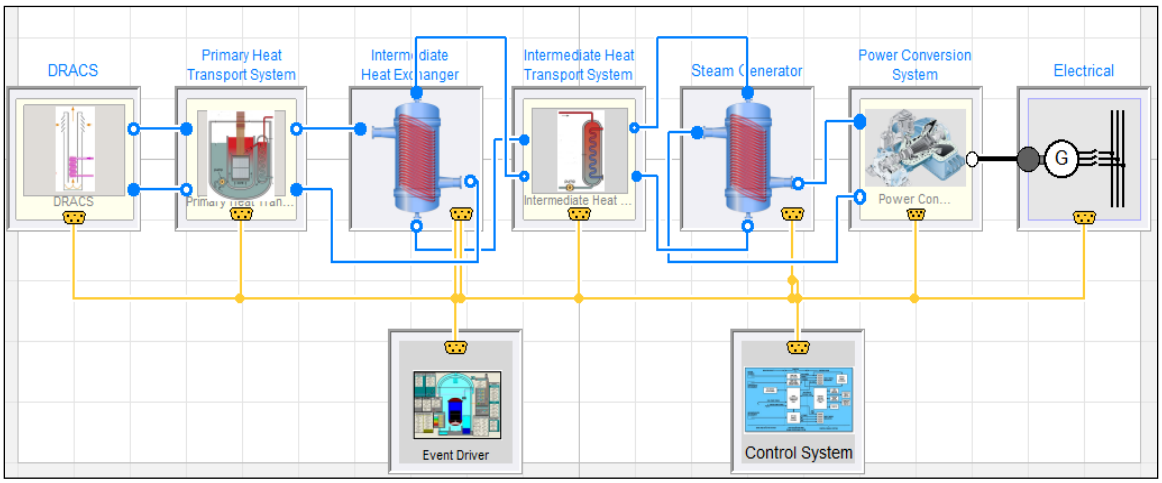


Fig. 1. Modular architecture for advanced liquid metal reactor end-to-end system model.

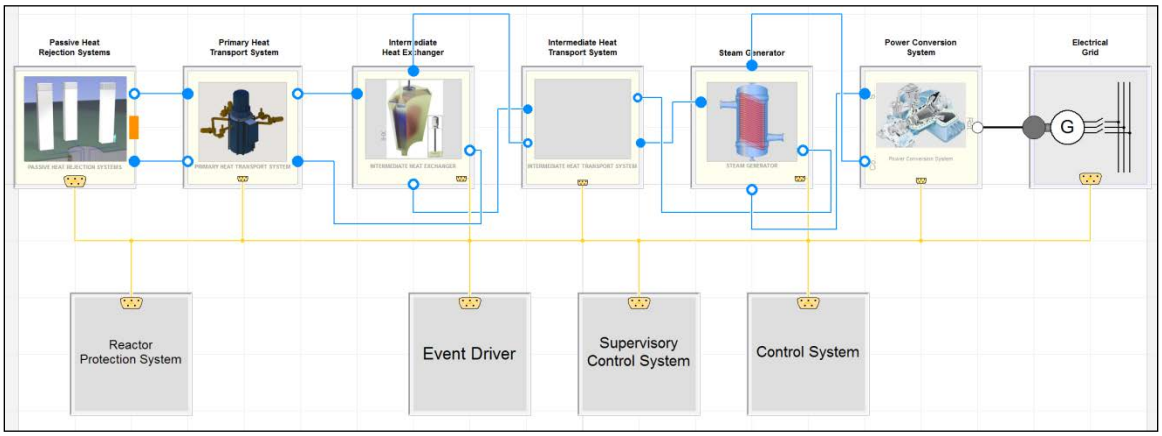


Fig. 2. Modular architecture for fluoride high-temperature reactor end-to-end system model.

2.2.1 Web-Based Application Deployment/Hosting

The server/hosting requirements for the web-based application are minimal. The application is being deployed as a collection of “Docker” containers.

Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. The Docker suite consists of the Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows. Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, data center VMs, and any cloud [4]

Docker was chosen as the deployment platform for a number of reasons. First, it significantly reduces the overhead associated with virtualization while preserving much of the reliability and safety. Second, there are a number of providers for Docker hosting which is an important factor in reducing the need for special hardware or potentially even any hardware.

Docker is a system that allows different (Linux) OS installations to be hosted on a single machine with almost no memory or disk storage overhead. All virtual machines share memory and disk space but are sufficiently partitioned to avoid any mixing of data (i.e., to maintain security). This has the added benefit that the same machine (and in some sense the same Linux host) can have multiple versions of the supporting software tools installed as long as they exist in separate containers.

Docker provides what amounts to a firewall between every virtual machine. This makes it very easy to relocate Docker images to other hosts. The use of Docker means it can be almost trivially relocated between different hosting environments, and the program can be run on most server environments. It even can be run on a Windows desktop running tools like VMWare or VirtualBox (although it isn’t clear what impact this would have on performance).

2.2.2 Web-based Application Authentication and Authorization

The release of a beta version of the web-based application necessitates the verification of a potential user’s credentials to access the application. The goal is to ensure access to only those participants that have been approved and authenticated. Collaboration for model development is facilitated with GitHub, and GitHub-based (OAuth) authentication is provided as an option to users of the web-based application. This easily handles the case of GitHub developers but does not necessarily include others, who may not be collaborating on model development. For this more casual user, a simple method of accomplishing the same level of authentication is to use existing authentication platforms that exist for widely used services. These include services like Google and Facebook. The objective of this approach is to use other authentication services without having to build in the software overhead in the application. The web application sysadmin can input the potential user’s GitHub, Google, or Facebook profile information, and the authentication is then passed through these services.

2.2.3 Web-based Application Structure/File Needs

The web-based application makes use of FMU files that are generated when the Modelica models are compiled. As described in previous reports (Refs. 1–3), FMI defines a standardized interface to be used in computer simulations of complex cyber-physical systems. The FMI implementation enables the creation of an FMU, either a model that can be interconnected or a self-contained simulation model with an integrator. A FMU can be used for simulations outside the native development tool. The web-based application takes the FMUs generated and connects these to the FMI-Add-In Tool for Excel provided by

Modelon, Inc. Therefore, the simulation environment for the web-based application is Excel. However, the use of the FMI Add-In also requires the establishment of parameters established for input and output. The web-based application automates this process. The previous prototype made use of separate files necessary to establish these parameters. The current beta version has progressed to a point where most information required to generate the web-based application and establish the simulation parameters is stored in the Modelica models directly. As a result, this approach has significantly streamlined the process of application generation and eliminated the previous (multistep) YAML-based approach. The new approach requires far less work and expertise from the modeler, and it is a completely repeatable process so it is easy to keep the web-based applications synchronized as the models evolve. The key information required for the web-based application includes:

- architecture to which a model belongs,
- subsystems within an architecture,
- subsystem implementations used in a particular model, and
- parameters and outputs associated with a given subsystem.

The new approach has the following benefits:

- elimination of YAML files,
- annotation of models with information to support web app creation,
- creation of FMUs from annotated models, and
- Xogeny web application built from FMUs by parsing model annotations.

Annotation of the models to be used by the web-based application follows the following conventions:

- Parameters to be included in the web-based application should be propagated to the top level of a subsystem.
- Outputs should be defined as output variables at the top level of a subsystem.
- Parameters and outputs can be defined in base classes plus actual implementations.
  - Common parameters and outputs can be defined in base classes, while those specific to an implementation can be added in the implementation.
- Variable descriptions are used by web building applications to identify parameters and outputs.
- Subsystem “roles” must be unique (i.e., reactor, coolant loop, intermediate coolant loop, controller).
- Subsystem “choice” must be unique and thus should be assigned at implementation (i.e., each implementation should be given a unique identifier).

Additional details about how models are annotated in order to provide the necessary information for the web-based application are included in Sect. 5.2. The information covers steps to annotate models using the same reactor demonstration models used previously to illustrate the YAML approach.

2.2.4 Simplified Web-Based Application Updating

The prototype web-based application was initially developed without consideration of the need to update it rapidly to incorporate model updates. To accommodate this need, the web-based application was designed to facilitate update in three steps: (1) initialization, (2) creation, and (3) preview. Following successful preview, the application is published to the server. Details of the commands for updating the application are found in Appendix A Sect. 5.1.

2.2.5 Integration with Central Modeling Repository

The prototype web-based application makes use of files resident on a local server. In a collaborative environment, these files will reside in a central repository. This central repository has been established as a private repository in GitHub (Sect. 3.2.1). The beta version of the web-based application links to the necessary files in GitHub to further reduce the number of steps necessary to update and publish the web-based application for online use (Sect. 2.2.4).

2.2.6 Simplified User Interface

The prototype web-based application made use of drop down selections for web models to create an end-to-end system. One limitation, however, was the need for users to understand which potential drop down selections corresponded to available end-to-end system models. The beta version eliminates this problem. Each user-selected choice now serves to populate the remaining user choices only with available choices that constitute established end-to-end systems. This approach ensures that the user can not select a series of choices for which data does not exist. This approach also simplifies the user's interface and ensures the generation of usable data from the application. Where choices are unavailable, notes are provided to the user to identify needed data and points of contact for further development of desirable architectures.

2.3 WEB-BASED APPLICATION MODELS

The web-based application is designed to generate simulations from established end-to-end system FMUs. Individual component and subsystem models are connected via the architectures seen in Figs. 1–2. These component and subsystem models represent attempts at optimized designs to achieve the desired overall end system performance goals. A principal challenge is integrating individual component and subsystem designs into a simulatable overall plant architecture and providing the necessary controls to achieve plant performance. The difference between the advanced liquid-metal reactor (ALMR) and the fluoride high-temperature reactor (FHR) development of end-to-end system models serves as a good example and is described later.

2.3.1 Liquid Metal Reactor Models

The liquid metal reactor models are based on the power reactor innovative small module (PRISM) design seen in Fig. 3 and described in detail in Ref. 2. The ALMR model is the most complete end-to-end system developed using Modelica to date. The design layout and architecture for this end-to-end system model is seen in Fig. 4. This work was based on a full design documented in Ref. 5. As a result, the individual component models and end-to-end system models have data for which the Modelica models can and have been calibrated. The subsystem and component models used in the web-based application to represent the ALMR architecture are described in Sect. 3.3.

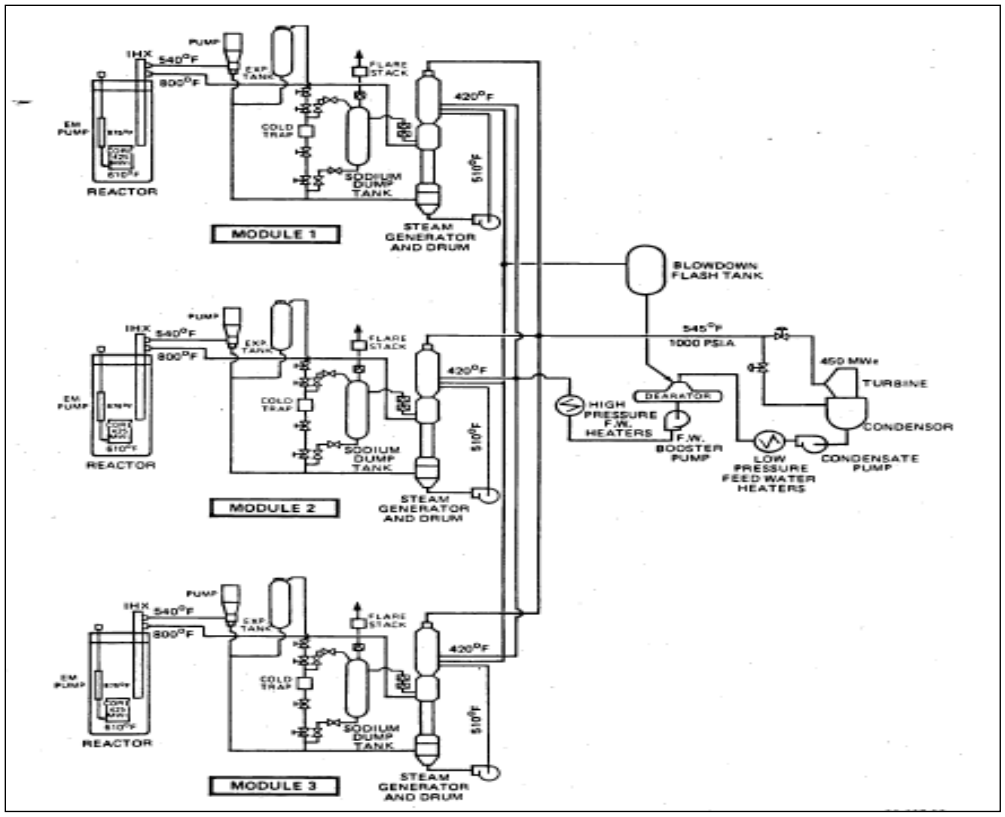


Fig. 3. PRISM system design.

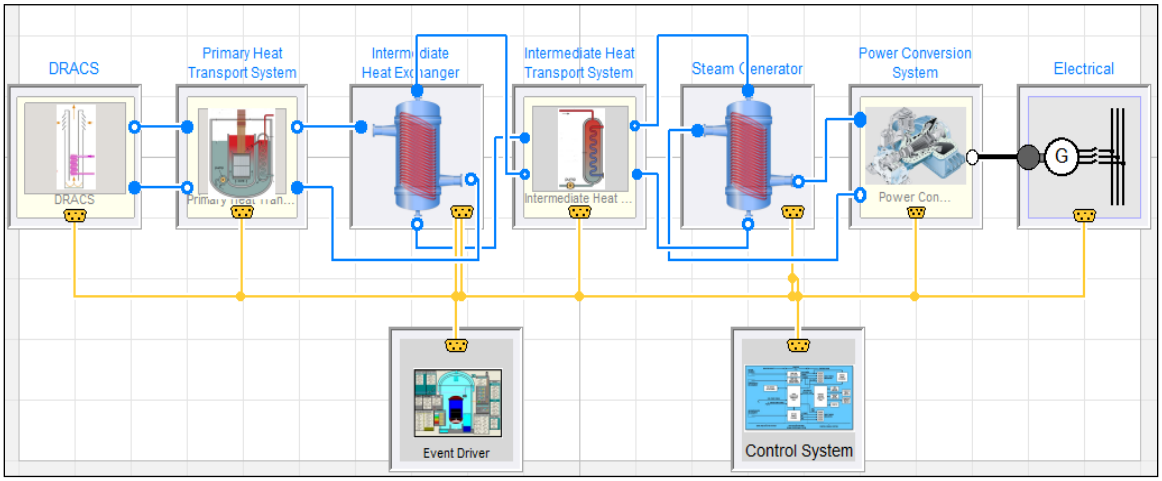


Fig. 4. Modular architecture for advanced liquid-metal reactor end-to-end system model.

2.3.2 Fluoride High-Temperature Reactor Models—Update

The FHR models are based on the small modular advanced high-temperature reactor (SmAHTR) design shown in Fig. 5 and described in detail in Ref. 3. A description of the current state of the end-to-end FHR model is included below. The FHR model is conceptual and therefore has not been optimized for all subsystem and components. The final architecture for the FHR end-to-end system model is shown in Fig. 6. The currently implemented architecture for the system is shown in Fig. 7. This work was based upon a conceptual design documented in Ref. 6. As a result, the individual component models and end-to-end system models do not have data by which the Modelica models can be calibrated. This represents a challenge when the components and subsystems are integrated to develop end-to-end models for the web-based application. The subsystem and component models that are used in the web-based application to represent the FHR architecture are described in Sect. 3.3.

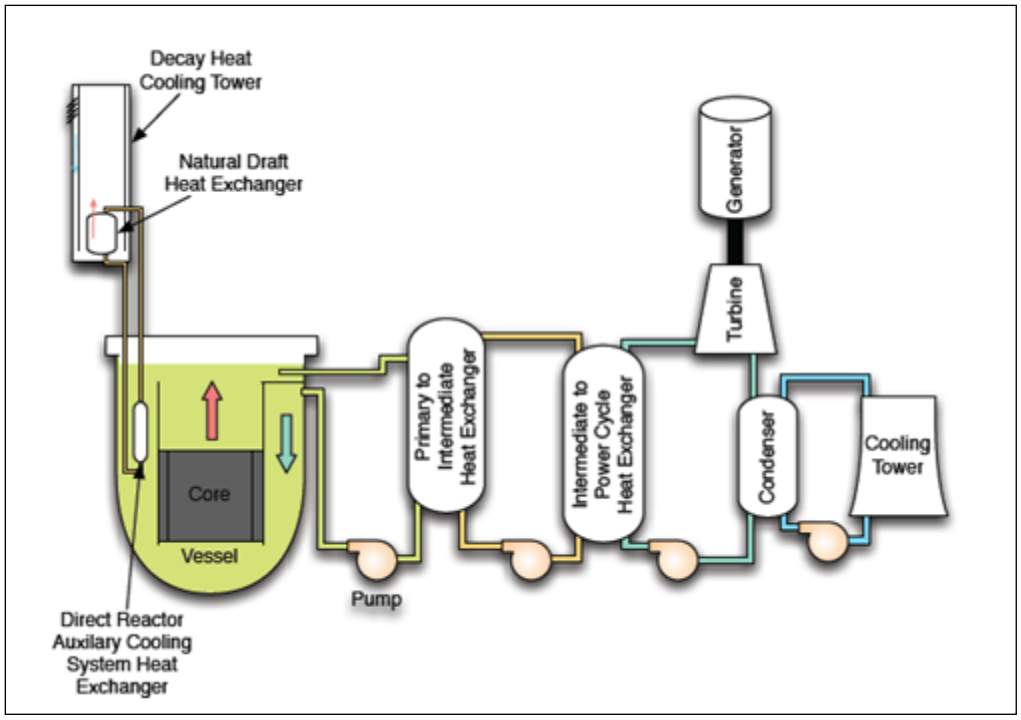


Fig. 5. Small modular advanced high-temperature reactor system design.

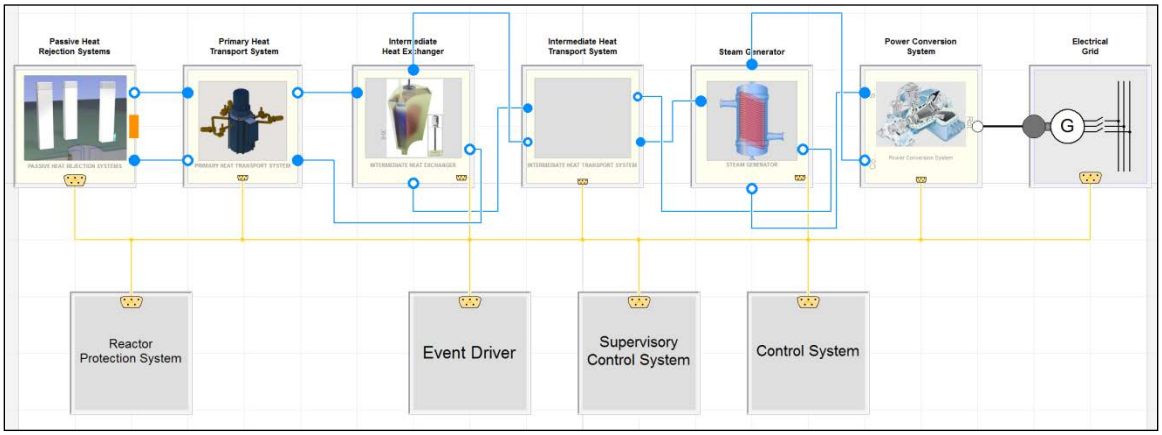


Fig. 6. Modular architecture for fluoride high-temperature reactor end-to-end system model.

Oak Ridge National Laboratory (ORNL) previously delivered a Modelica model of the SmAHTR plant. While this model met the performance requirements at the individual subsystem level, the integrated end-to-end system model was not complete. The SmAHTR design is still conceptual, and unlike the ALMR PRISM plant designs, detailed component and interface definitions are incomplete. ORNL conducted limited design revisions based on plant functional requirements and high-level performance specifications such as the amount of steam to be delivered.

The top-level diagram of the SmAHTR plant model is shown in Fig. 7. The model includes the primary heat transport system (PHTS), the intermediate heat exchanger (IHX), and the intermediate heat transport system (IHTS). Since the IHTS and the steam generator designs are not available, the IHTS performance is calculated using mass flow and enthalpy boundary conditions at the steam generator interface.

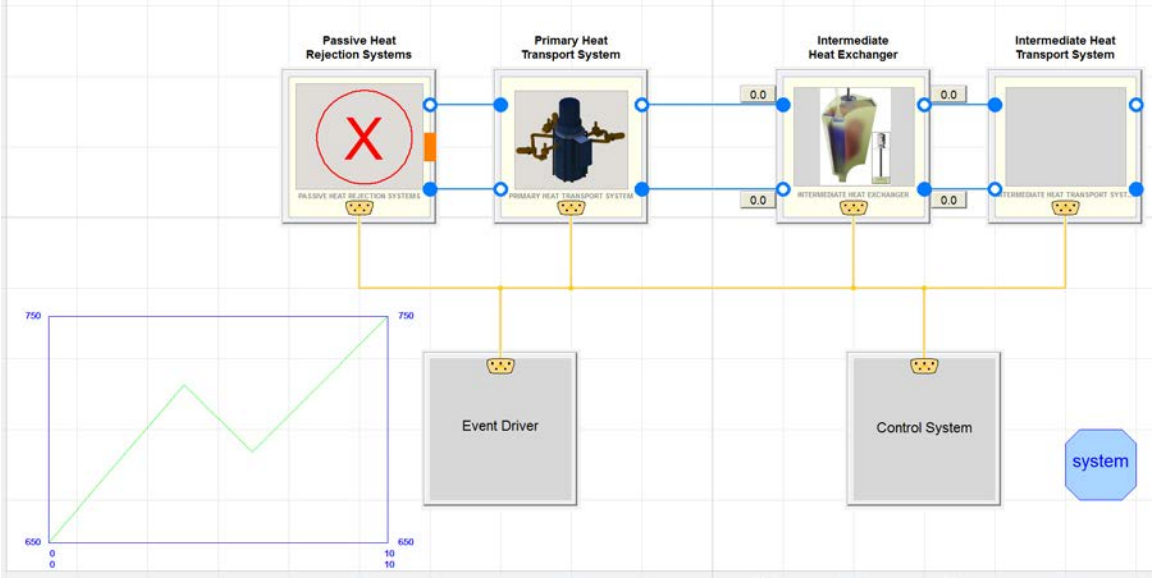


Fig. 7. Current top-level integrated end-to-end system model of the small modular advanced high-temperature reactor plant.

The SmAHTR employs a direct reactor auxiliary cooling system (DRACS) safety-grade passive heat rejection system for dissipation of decay heat during an extended station blackout condition. However, the scope of this dynamic simulation project is currently limited to anticipated operational occurrences. ORNL created a null subsystem block to allow incorporation of future implementations of the DRACS. The key design parameters of the SmAHTR are reported in Ref. 3. The same core, fuel, and interconnected coolant system parameters were in this model. The IHX design parameters were modified slightly to match the performance specifications such as power output per IHX, coolant flow velocity within the channels (shell and tube side), and pressure drops across.

2.3.2.1 Primary heat transport system

Two versions of the PHTS model were created to demonstrate the power and modularity of the simulation package. Both models consist of a coolant channel coupled to the fuel element; a primary pump; a lower plenum; and an upper plenum, where fluid mixing occurs. The single-zone core model includes one reactor module, connected to a fuel element and a coolant channel as shown in Fig. 8. This implementation is the baseline implementation.

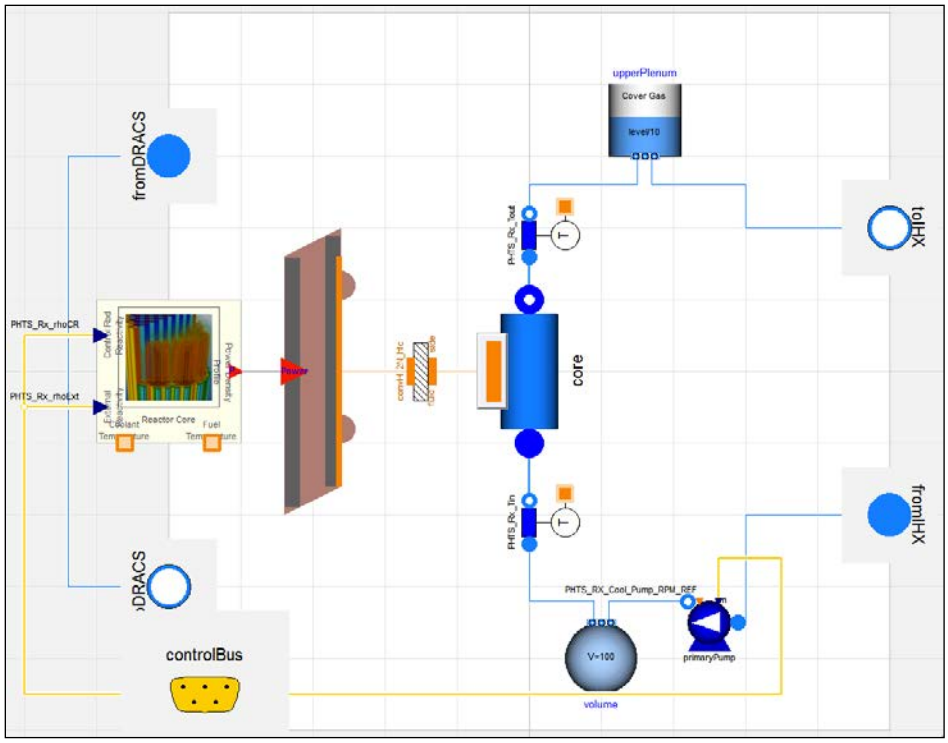


Fig. 8. Single-zone model of the reactor, fuel, and the connected coolant system.

The two-zone core model includes two reactor modules, each connected to their fuel element and coolant channel as shown in Fig. 9. This implementation provides a basic capability for simulation of radial power profile distributions. Each reactor is furnished with its own reactivity control elements. This implementation also provides the capability to demonstrate multivariate control. Currently, the two reactors are not neutronically coupled.

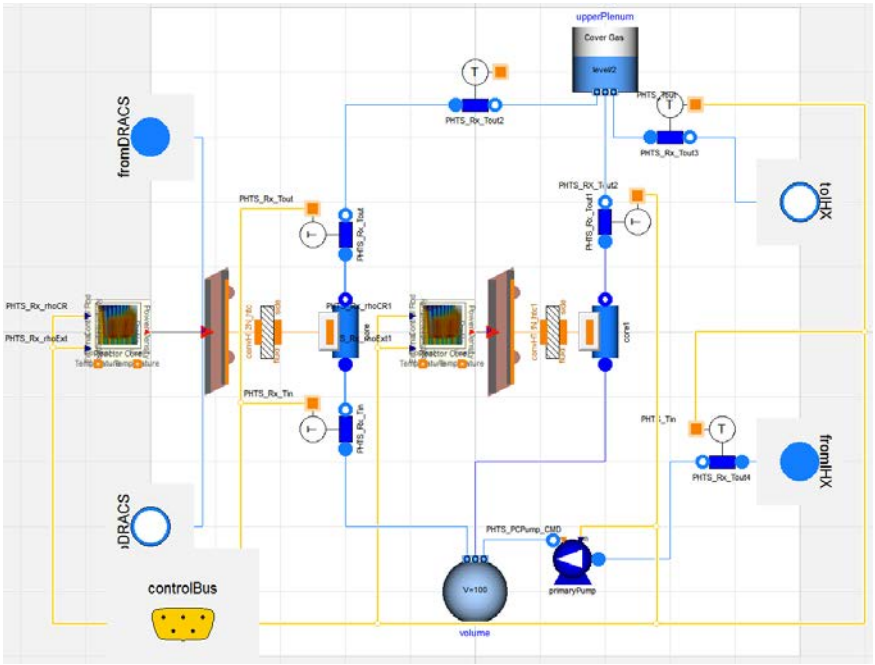


Fig. 9. Dual-zone model of the reactor, fuel, and the connected coolant system.

2.3.2.2 Intermediate heat exchanger

The SmAHTR IHX model is nearly identical to the one reported in Ref. 3. Minor modifications were made to match the hydraulic characteristic of the design. The Modelica diagram layer of the IHX is shown in Fig. 10.

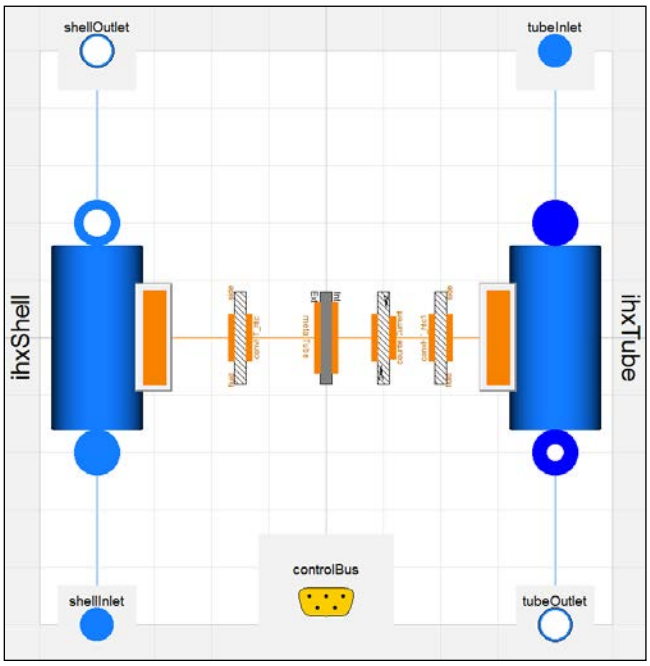


Fig. 10. Modelica diagram layer of the small modular advanced high-temperature reactor intermediate heat exchanger model.

The primary fluid (flibe) flows on the shell side while the secondary fluid (flinak) flows on the tube side. Nominal steady state temperature profiles on the shell and tube sides are shown in Fig. 11

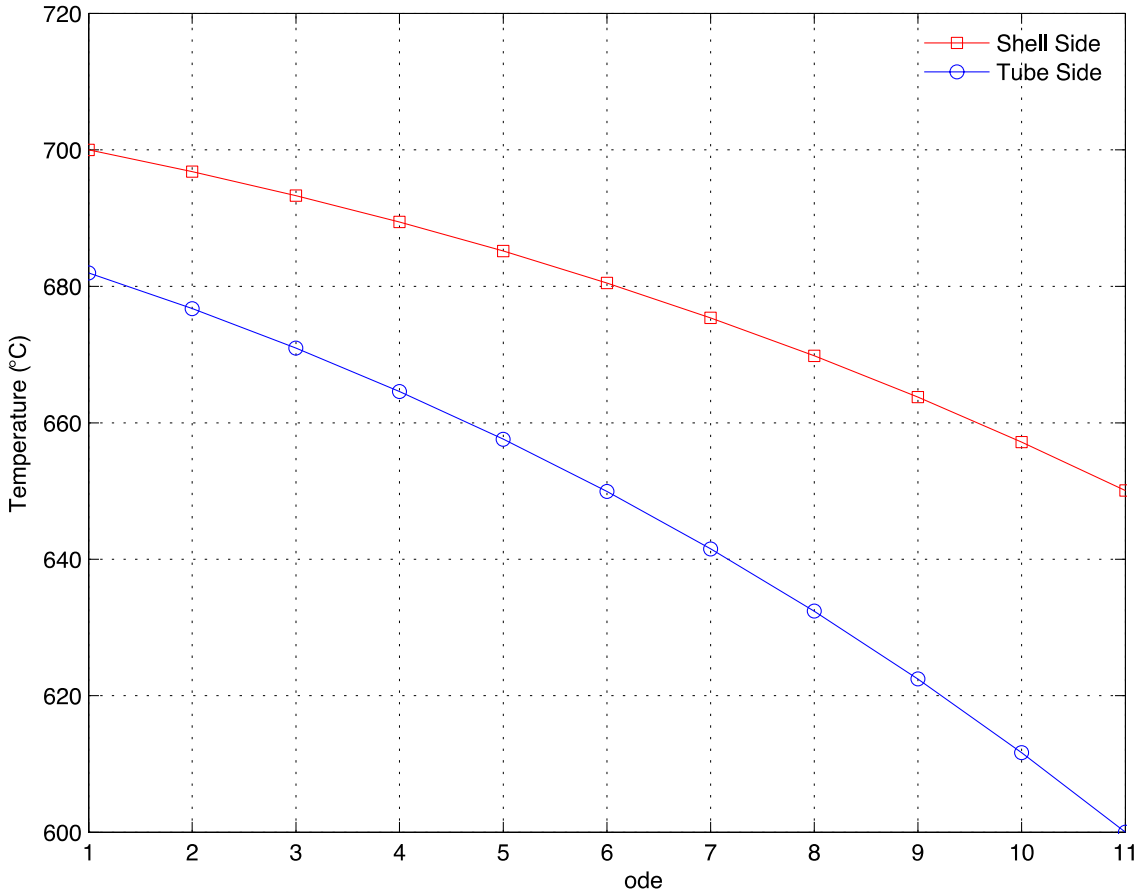


Fig. 11. Temperature profiles on the shell and tube sides of the small modular advanced high-temperature reactor intermediate heat exchanger.

2.3.3 Intermediate Heat Transport System

The IHTS couples the IHX and the steam generator (Fig. 12). The length of the piping each way accounts for the dynamic time delay between core power generation and steam generator heat rejection. As the SmAHTR steam generator design information is limited, the IHTS performance is assured by using boundary conditions at the steam generator interface.

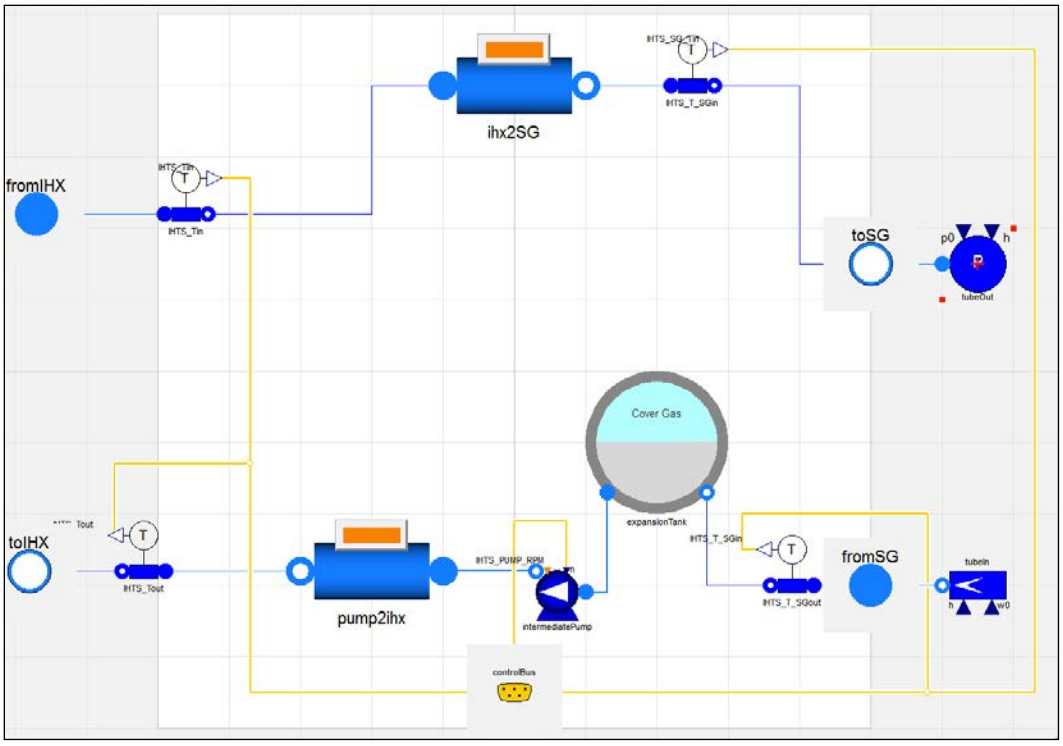


Fig. 12. Modelica diagram layer of the small modular advanced high-temperature reactor intermediate heat transport system.

2.4 INSTRUMENTATION AND CONTROLS OVERLAYS AND SIMULATION

The Modelica models are designed for evaluating potential control concepts for small modular reactors and advanced reactor concepts under transient operational conditions. To accommodate this within the Modelica architectural framework, separate modules for instrumentation and controls (I&C) overlays and event drivers have been developed. To date, two control system strategies have been implemented and demonstrated. The Modelica structure and flexibility for these two control strategies are displayed in Figs. 13–14. The first control strategy is based on controlling to temperature setpoints at key locations in the plant model; the second control strategy is based on controlling to temperature difference setpoints at key locations in the plant model. These control strategies are implemented through the conceptual architecture as displayed in Fig. 13, with the modeling implementation of the architecture as displayed in Fig. 14. Both strategies are fully developed and discussed for the ALMR design in Ref. 2 and have been implemented on the FHR concept.

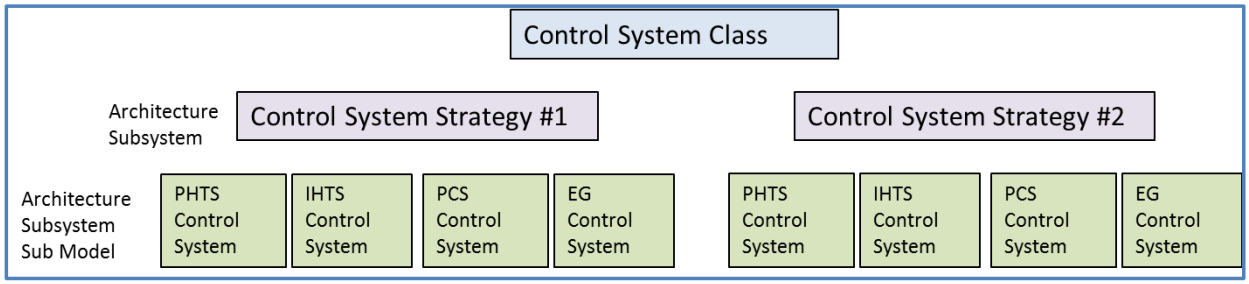


Fig. 13. Instrumentation and controls conceptual diagram.

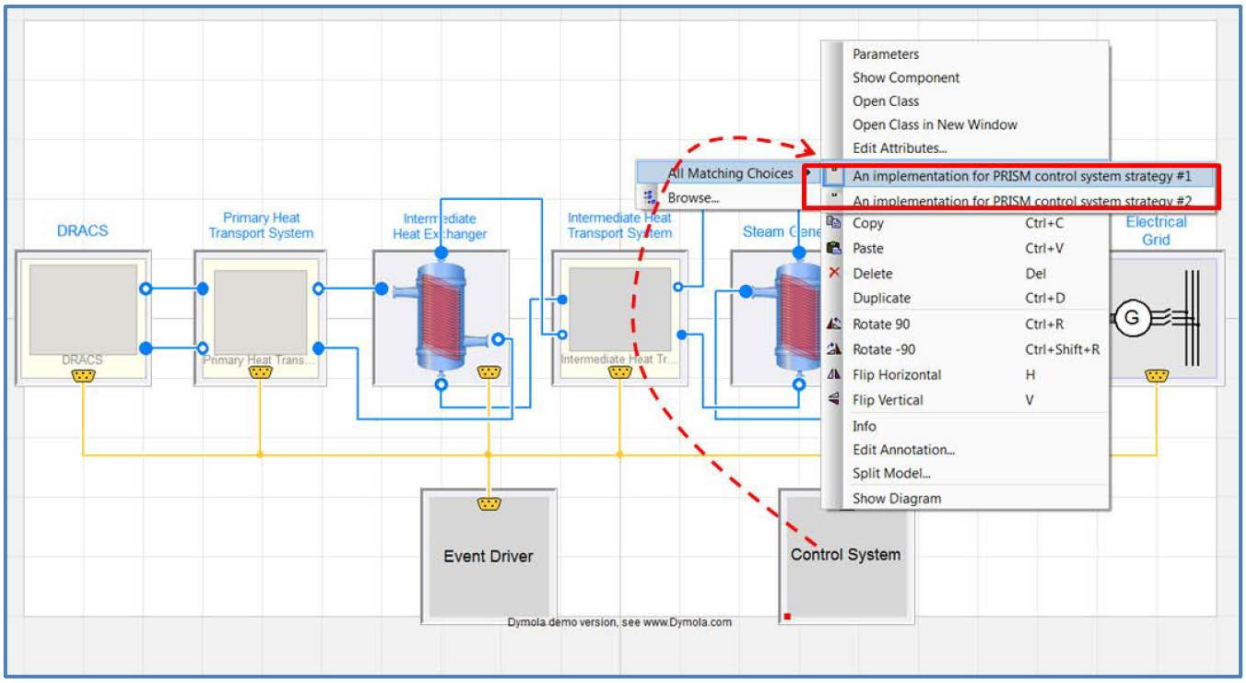


Fig. 14. Instrumentation and controls Modelica diagram.

The baseline FHR model contained PHTS, IHX, IHTS, event driver (ED), and control system components. The model was updated to use appropriate molten salt thermophysical properties. The SmAHTR conceptual design information is less mature than the PRISM example, which results in less system and subsystem design details to develop the model components.

PHTS and IHTS were updated to add temperature measurements and control to support I&C modeling. The PHTS reactivity control rod and primary cooling pump and the IHTS cooling pump were added to the control bus for control system access and connectivity. The IHTS cooling pump model has not yet been properly developed with a suitable pump curve due to the limited SmAHTR reference information.

Control strategy #1 for the FHR model is developed as position control loops for the PHTS reactivity control rod and the PHTS primary cooling pump speed (Fig. 15). The reactivity control rod controller is a proportional-integral control loop that adjusts the reactivity control rod position to maintain the reactor outlet temperature to a setpoint of 700°C. The primary cooling pump speed controller is a proportional-integral control loop that adjusts the pump speed to maintain the PHTS inlet temperature (return from the IHX) to a 650°C setpoint.

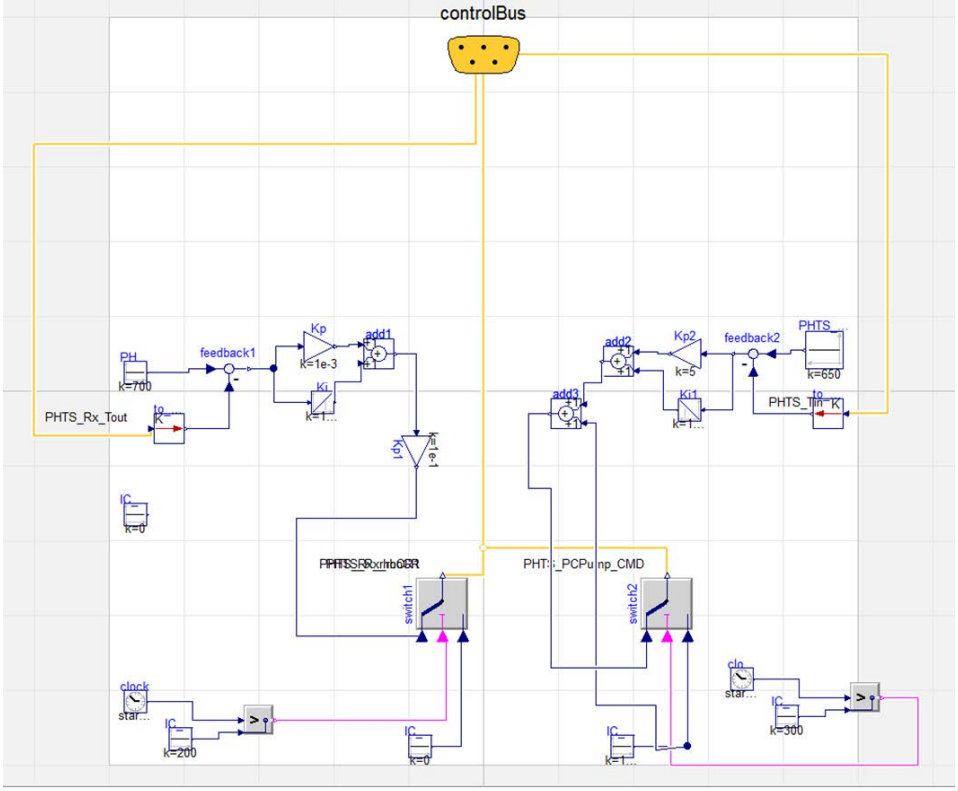


Fig. 15. Control loops for primary heat transport system reactivity control and primary cooling pump.

2.4.1 Fluoride High-Temperature Reactor Example Transient Simulations

The response of the FHR system to system disturbances was examined. In Fig. 16, a reactivity disturbance step change occurs at 1500 seconds (noted as event #1 on the figure). The reactivity control rod control can be observed making a correction to maintain the reactor outlet temperature, and the PHTS primary cooling pump control makes a transient correction to reduce the upset to the PHTS inlet temperature. The IHTS cooling pump and fluid flow is fixed speed (not a control variable). In Fig. 17, a 13% increase in flow disturbance in IHTS occurs at 1500 seconds. The PHTS reactivity control and the PHTS primary pump control increase the reactivity and the pump speed to maintain the desired temperatures.

2.5 WEB-BASED APPLICATION USE

The ALMR end-to-end model is complete. The current state of the end-to-end FHR model is described in the previous section. As described in Refs. 1–3, the development of high-fidelity models is a deeply technical undertaking requiring highly specific skills. However, the use of the resulting models for analysis is open to a much broader set of participants. To make these models accessible to others, it is important to have one or more avenues through which the models can be "deployed." This deployment should move the models out of the model development tools needed to create them and into forms that are intuitive and easy to use.

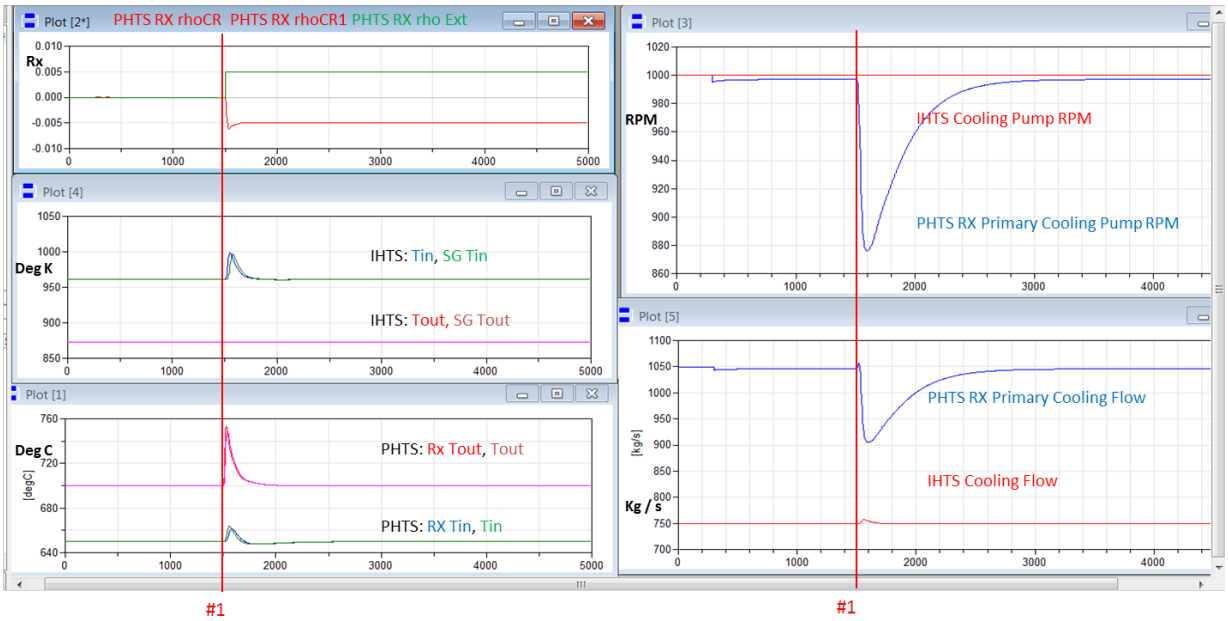


Fig. 16. Reactivity disturbance example simulation.

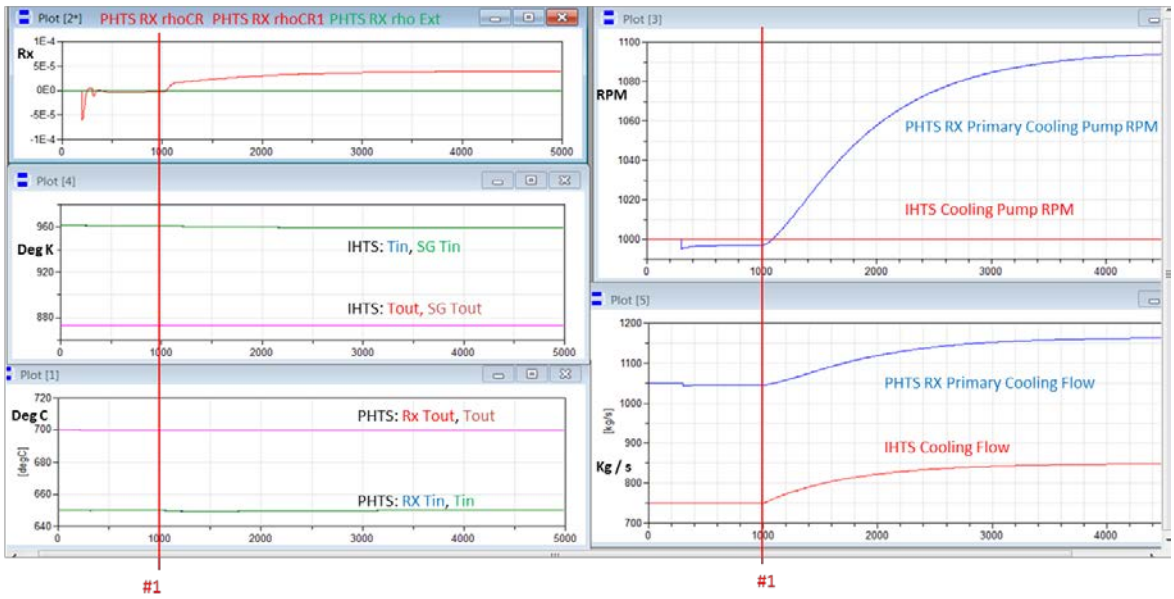


Fig. 17. Intermediate heat transport system flow disturbance example simulation.

For this reason, we are working with Modelon and Xogeny to develop web-based engineering analysis tools for our models. These tools take the models (compiled into FMUs) and transform them into web-based applications for each architecture of interest.

An important feature of this process is that these applications are generated in a virtually automatic way. By leveraging information contained in the models themselves, the Xenarius application generator (developed by Xogeny) is able to create complete web applications without writing any code.

Upon startup, the user is given a choice of different architectures to explore, as shown in the screenshot below.

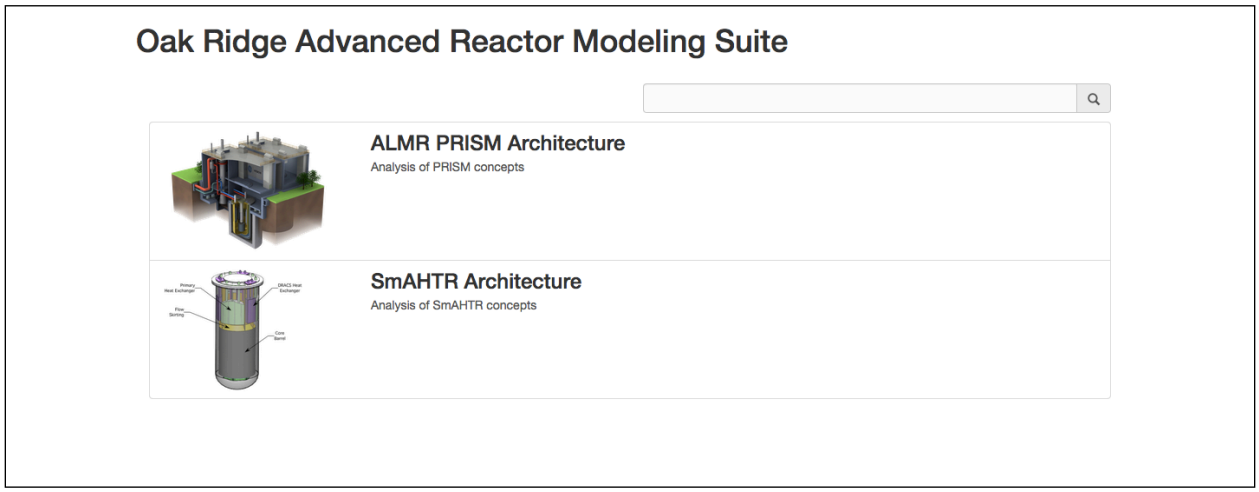


Fig. 18. Web-based application reactor architecture selection.

From this screen, the user selects the architecture of interest. At present, only two architectures are supported. But as this list grows, the search bar shown can help prune the set of architectures.

Once the architecture has been chosen, the user is presented with configuration options for the various subsystems in that architecture (Fig. 19).

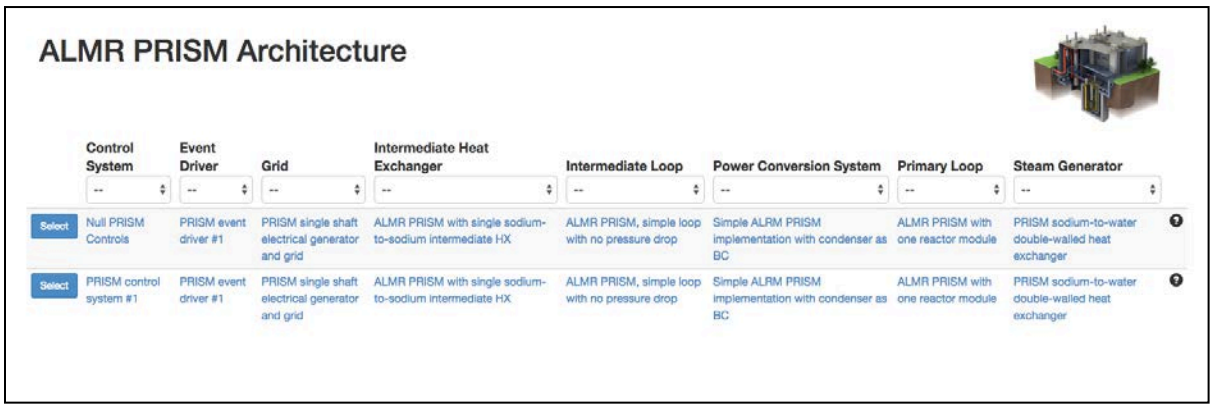


Fig. 19. Web-based application subsystem architecture selection shown in a screen capture from a desktop browser.

This screen allows the user to filter specific configuration choices to find a model that fits the system configuration he is interested in. The list of system models is filtered based on specific subsystem requirements.

Once a particular configuration is chosen, the user is presented with the key design parameters associated with that configuration as indicated by the model developer in the annotated models. These parameters can be edited to formulate specific designs. A complete set of design parameters is available to the advanced user via the Excel interface.

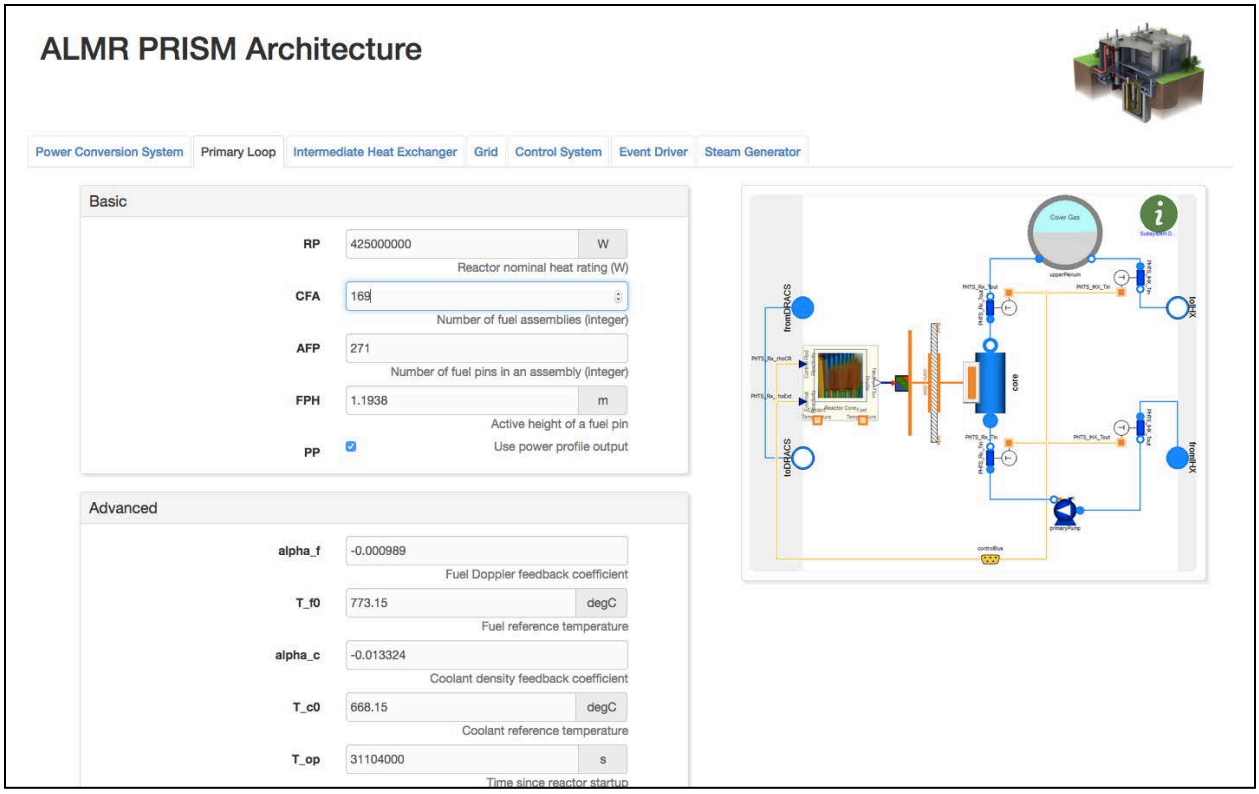


Fig. 20. Web-based application subsystem parameter selection.

These designs can then be downloaded as Excel spreadsheets and simulated via FMI Add-in Tool for Excel from Modelon. The screenshot below shows the page with the Excel sheet download.



Fig. 21. Web-based application Excel download (FMI Add-in).

The Excel sheet served by the web app is fully automated to download the FMU with the design, simulate, and plot variables annotated by the model developer as key outputs for each subsystem. The screenshot below shows the user interface when the Excel sheet is opened. Clicking "Download Remote FMU" will download, simulate, and plot with one click.

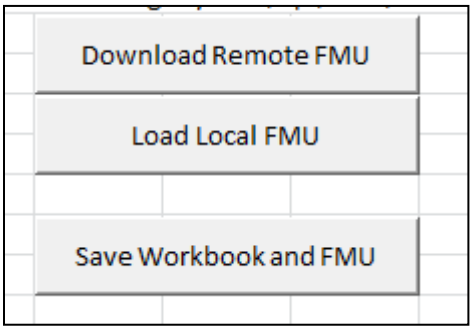


Fig. 22. Web-based application Excel option (FMI Add-in).

The automated worksheet also creates an experiment sheet which can be used for additional batch simulations via further changes in the design parameters.

Model														
Sheet version	Generated by Modelon FMI Add-In for Excel version 1.3.3													
Model name	TRANSFORM.PlantSystems.ALMR_PRISM.Rx2Grid2_CS1													
Generation tool	Dymola Version 2015 FD01 (64-bit), 2014-12-15													
FMU kind	CoSimulation_StandAlone													
Number of processes	8													
Checksum	143feef677f44b5352b4666aade15b3c													
Expiry date														
Settings				Default	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10
Start time				0										
Stop time				5000										
FMU				C:\Users\jbatteh\Documents\Work\ORNL_2015\AdvSMR\Scripts\TRANSFORM_PlantSystems_ALMR_0PRISM										
Log level				Info										
Enable				TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Output points				100										
Timeout				0										
Indata														
Name	Variability	Type	Unit											
primary_loop.Q_nom	parameter	Real	W	4.3E+08										
primary_loop.noFuelAssemblies	parameter	Integer		169										
primary_loop.noFuelPinsPerAssembly	parameter	Integer		271										
primary_loop.H_fuelPin	parameter	Real	m	1.1938										
primary_loop.usePowerProfile	parameter	Boolean		TRUE										
primary_loop.alpha_f	parameter	Real		-0.00099										
primary_loop.T_f0	parameter	Real	K	773.15										
primary_loop.alpha_c	parameter	Real		-0.01332										
primary_loop.T_c0	parameter	Real	K	668.15										

Simulate and Plot
Open Plot GUI

Fig. 23. Excel experiment sheet (FMI Add-in).

The parameters set in the web application are populated in the "Default" column of the experiment sheet and are simulated and plotted automatically. A subset of the plot output is shown below. All output variables annotated in the model are plotted. The user can run batch experiments in parallel using all the cores on the local machine by simply enabling the additional cases and providing the parameter values to be simulated. Users can also add additional parameters and outputs to the experiment sheet to allow full access to the design for advanced users (i.e., beyond those annotated in the model by the developers).

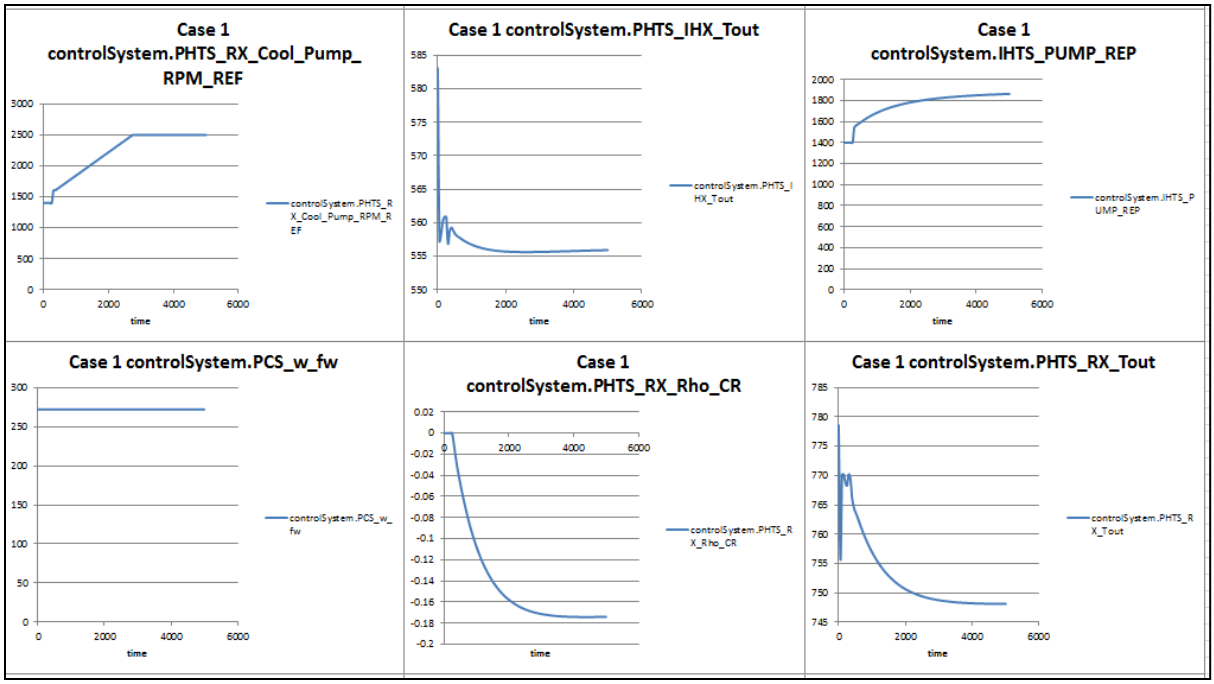


Fig. 24. Automatic Excel plot generation (FMI Add-in).

Another advantage of the approach of annotating the parameters and outputs in the model is that the Excel sheet automation can be used for any model, even without the web application. Any users with access to the model source code in GitHub can simply generate an FMU from the annotated models, load the FMU into the Excel sheet via the "Load Local FMU" option, and run simulations with all outputs automatically plotted. The automatically created experiment sheet can be used for further design studies and to conduct batch simulations.

2.6 FUTURE WEB-BASED APPLICATION DEVELOPMENT

2.6.1 Online Simulation

In the near future, we expect this web-based application to also allow completely interactive, web-based analysis without the need for external tools. While simulation from the web application in Excel offers the most flexibility for more technical users who wish to "crunch the data," a web-based analysis tool that allows designs to be explored completely through the web opens the door to many additional possibilities. Users are no longer tied to a specific platform or tools. All they need is a web browser. This allows architecture studies to be performed anytime, anywhere and on any device.

The current application requires the use of Excel to perform a simulation through the web application. Although this approach gives great flexibility for simulation in an environment that most engineers are quite familiar with, it does require a user to have Excel and access to a license for the FMI Add-in tool. The currently anticipated workflow involves the use of a shareable license that will be served via FlexLM from the web application server. This licensing mechanism will allow any authorized user to access the license and simulate models using his local version of Excel. Additional licenses for the FMI Add-in held separately by users would eliminate any potential conflict for multiple users accessing the application simultaneously. Additional sharable licenses can also be served from the web server to meet anticipated simulation demand. An additional workflow will allow the users to simulate directly on the Xogeny platform. This approach would eliminate the need for tool-specific licensing (though some sort of

licensing mechanism via the Xogeny platform is anticipated) and would enable simulation directly through the browser. To underscore what possibilities exist under this workflow plan, consider the configure selection screen from the web-based application.

Figure 19 is a screen capture of a configuration selection screen from a desktop browser. Figures 25 and 26 are screenshots of the same configuration selection screen taken from mobile devices (an iPad and an iPhone, respectively):

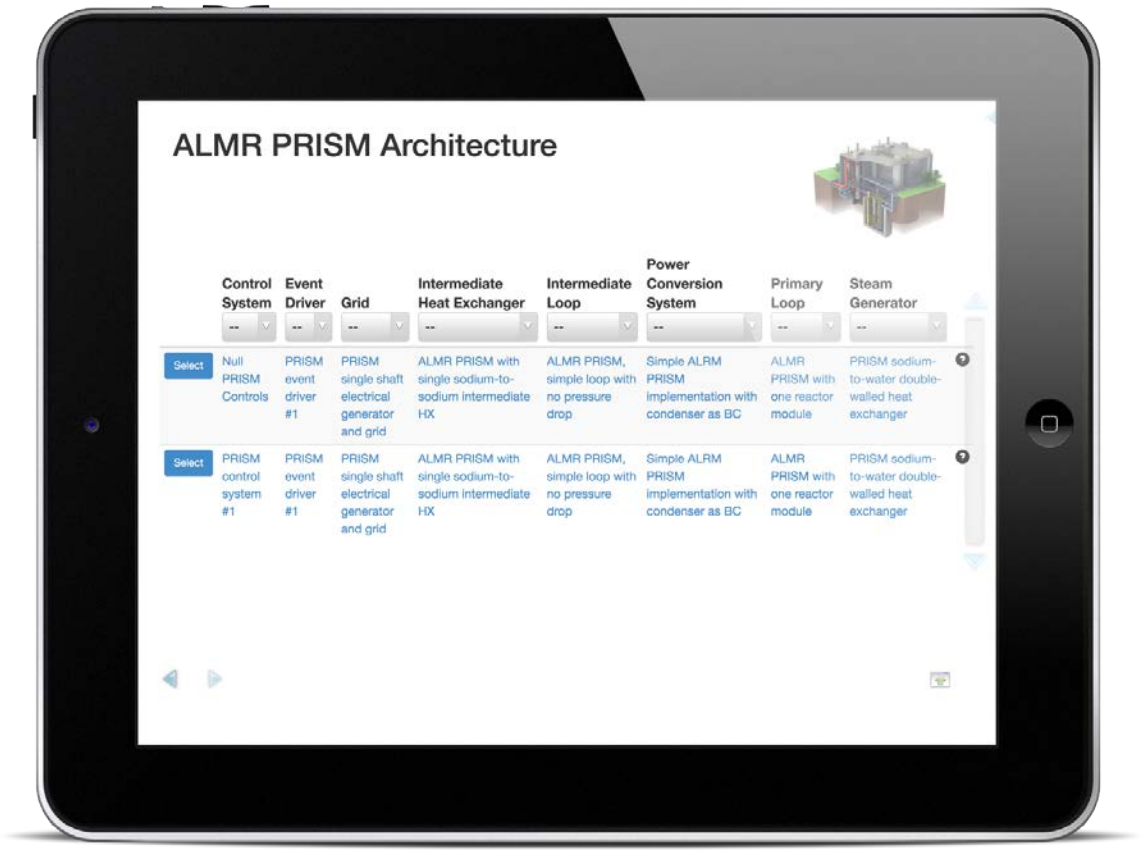


Fig. 25. iPad display for subsystem architecture selection.

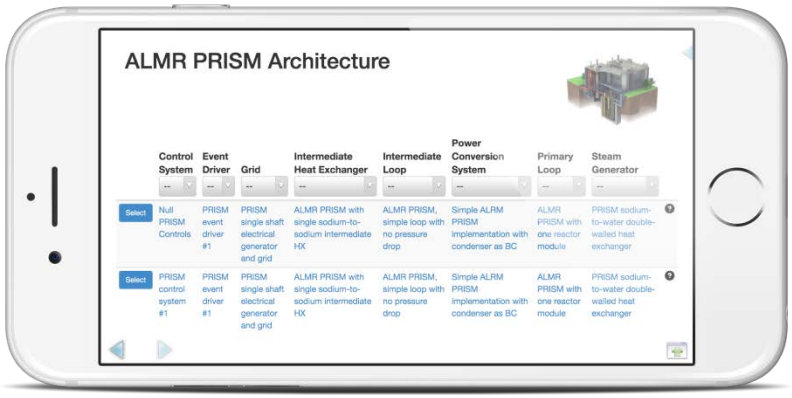


Fig. 26. iPhone display for subsystem architecture selection.

Likewise, Figs. 27 and 28, screenshots of the Web-based application subsystem parameter selection on an iPad and an iPhone, illustrate that this subsystem is completely usable from a mobile device or from a desktop browser (Fig. 20).



Fig. 27. iPad display for subsystem parameter selection.

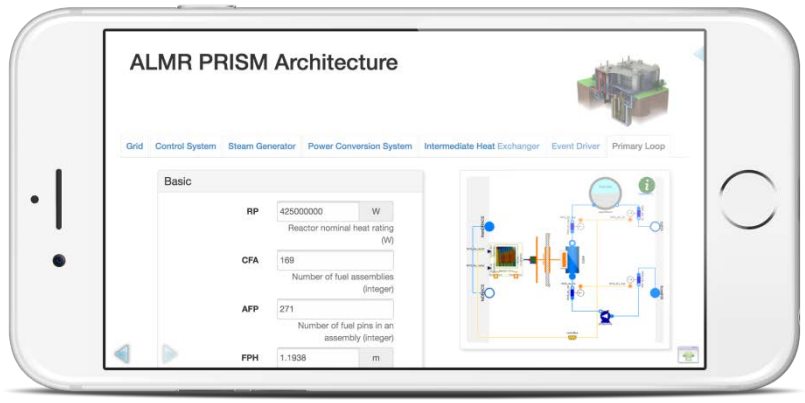


Fig. 28. iPhone display for subsystem parameter selection.

The key point is that the technologies being used here support what is called “responsive design.” This is a term to describe web applications that adjust based on the resolution of the screen used when accessing them.

In other words, the development of this web-based tool has not only successfully created a desktop analysis tool that is accessible by anybody on a network, it has also made it possible (without having to develop a separate iOS or Android version of the application) for users to perform architecture studies of various reactor configurations on any web-enabled device.

2.6.2 FMI Development

The current implementation of the web-based application produces an Excel sheet for simulating the model after the user configures the model and sets parameters using FMI Add-in for Excel from Modelon. The Excel-based simulation approach works in conjunction with the web app to deliver off-line model simulation and analysis capability based on the FMI standard. There are some significant advantages to FMI-based simulation in Excel, especially for users who need more capability than can be reasonably implemented in online simulation focused on a quick start for novice users. These features include the following:

- Experiment sheets to easily set up batch simulations and save/modify parameter settings.
- Ability to save customized, local sheets with custom parameter settings, post-processing/analysis, etc.
- Native utilization of all machine local cores for parallel processing.
- Automation with Visual Basic for Application (VBA) builds on same model annotations for web-based application to allow automated experiment sheet creation and plotting with automation built into sheets that can be distributed to local users (one click to download sheet with FMU and automatically run model and plot results, including comparison between batch runs).
- Ability for advanced users to access additional parameters and outputs that were not accessible via web interface (impossible to understand all use cases for all users via pre-compiled web-based application).

- Automation with VBA to provide additional user-defined front ends, analysis front ends, or even parameter setting GUIs in addition to what is provided via the web-based application.

- Additional analysis capabilities like Monte Carlo and Design of Experiments can be implemented directly by users.

- Native Excel interface for setting parameters, plotting results, and performing analysis on final values from simulation along with time histories.

- Simulation capability for any FMU generated by an FMI-compliant tool, which makes it possible for the program to be used as a generic model simulator and deployment platform without requiring a web-based application interface (workflow directly from a model development environment to a simulator in Excel).

- FMIE offers a local simulation and deployment platform that does not require online model storage since cloud-based options may not always meet security requirements.

- FMIE also offers a rapid development environment for simulation of models that are in the pre-deployment phase before web-based application deployment.

- Extension via VBA scripting using FMIE APIs.

Flexible licensing with standard FLEXlm technology allows license sharing and distinctions between model developers and run-time users for cost-effective deployment based on licenses as opposed to the number of simulations. To further enhance ORNL simulator workflows, future development of the ORNL infrastructure and FMIE tool will focus on the following:

- native analysis capability and post-processing for common analyses (Monte Carlo, Design of Experiments, Robustness Analysis, Design for Six Sigma);

- additional options for saving experiment sheet settings for rapid creation of experiment sheets without VBA scripting;

- additional licensing options for situations where license serving via a license server is not deemed appropriate due to security and license server integrity requirements; and

- additional compute options for distribution of simulations to local compute machines from FMIE front end.

3. MODEL COLLABORATION

3.1 INTRODUCTION/TASK DESCRIPTION

The use of the web-based application in concert with a private repository established through GitHub allows for the collaborative development and use of advanced reactor models within the DOE ART Program. The web-based application and the integrated end-to-end system models developed to date were presented in Sect. 2. The intention for collaboration is that multiple collaborators (government, industry, academia) will work together to develop component models that are used in the development of end-to-end systems.

3.2 COLLABORATION PLATFORM

The two collaboration platforms are the GitHub repository (for model development and sharing) and the ModSIM Web-based application (for analysis and results collaboration and sharing). A brief description of the Web-based application platform is provided in Sects. 2.2–2.3. GitHub is described in detail in Refs. 2–3. A specific GitHub repository has been established to serve as the development platform for models that are incorporated into the ModSim web-based application.

3.2.1 ORNL-ARM

ORNL-ARM is a private model repository established for advanced reactor models created by DOE and its contractors as well as established collaboration partners. This repository hosts the source code for models that will be deployed as FMUs via the web-based application. The current list of potential collaboration partners [3] includes the following:

Collaboration partner	Point of contact	Initial focus area/notes
North Carolina State University	Nam Dinh	Data-driven safety/licensing modeling using Modelica
New Mexico State University	El-Genk	Space-based liquid sodium reactor
Massachusetts Institute of Technology	Charles Forsberg	FHR modeling
Georgia Institute of Technology	Bojan Petrovic Chris Paredis	Modelica domain expertise
Ohio State University	Xiaodong Sun	Advanced modeling and instrumentation and control in multiphase flows

DOE Partnerships

Collaboration partner	Point of contact	Initial focus area/notes
Idaho National Laboratory	Humberto Garcia	Hybrid energy systems modeling

Business Partnerships

Collaboration partner	Point of contact	Initial focus area/notes
Modelon, Inc.	John Batteh	FMI development
Xogeny, Inc.	Michael Tiller	Web-based application

ORNL and its partners work together in teams using the repository towards the development of advanced component models that can be integrated into advanced reactor system models. The objective is to establish teams along the lines of the architectural elements for system models described in Sect. 3.3. These collaboration teams have not yet been established although folders for these collaborations have been established in the ORNL-ARM GitHub repository.

3.3 COLLABORATION MODELS

The end-to-end systems for the currently available models follow the architecture displayed in Figs. 1 and 2, respectively, for the ALMR and FHR designs. Although the details regarding the individual components and integration differ, these two designs have similar architectures. The models for these architectures are available for collaboration in the GitHub repository. A brief description of the major architectural system and component models is included in the subsections below, with source code for these models included in the identified folders in the GitHub ORNL-ARM repository. Further detailed descriptions for the ALMR and the FHR models as well as the analytical basis for these models can be found in Refs. 2 and 3, respectively. Collaboration is permitted on all of these models.

3.3.1 DRACS/RVACS Models

The DRACS/RVACS (reactor vessel auxiliary cooling system) models include a heat exchanger and piping used for natural convective cooling following shutdown. Their principal function is to transfer shutdown heat away from the core. The difference between RVACS and DRACS is that a DRACS heat exchanger is submerged directly in the primary coolant, while the RVACS heat exchanger is external to the reactor vessel. With DRACS, the heat exchangers that couple the system to core coolant are contained within the primary vessel rather than relying on conduction and radiation heat transfer to transport heat through the primary system vessel and into the air. The interfacial points for these subsystem models are the atmosphere and PHTS. Currently, the FHR model has a null implementation for the DRACS model. As the model develops, the source code for the DRACS/RVACS models for the ALMR and FHR designs will be included in the GitHub private repository and will be available for modeling collaboration.

3.3.2 Primary Heat Transport System Models

The current core model PHTS includes the core and the primary system piping. Their principal functions are neutronics control and transferring heat away from the core. The interfacial points for the primary heat transfer model are the DRACS/RVACS subsystems and IHX. The source code for all primary system models for the ALMR and FHR designs is included in the GitHub private repository; it is available for modeling collaboration.

Reactor core components models

The current core model is a component model within the primary coolant subsystem model. Two core models currently exist, one for ALMR and one for an FHR. The source code for these models is included in the GitHub private repository; it is available for modeling collaboration.

3.3.3 Intermediate Heat Transport System Models

The current core model IHTS provides the interface between primary and intermediate coolants Its principal functions are to provide this separation and to optimize heat transfer between these coolants. The interfacial points for the intermediate heat transfer model are the steam generator and the IHX component models. All source code for the intermediate system models for the ALMR and FHR designs is included in the GitHub private repository; it is available for modeling collaboration.

3.3.4 Heat Exchanger Components Models

The current core IHX component model simulates the heat transfer between the primary and intermediate coolant systems. Its principal function is to provide this separation; it also optimizes heat transfer between these coolants. The interfacial points for the IHX are the steam generator and the IHX component models. The source code for all intermediate system models for the ALMR and FHR designs are included in the GitHub private repository and are available for modeling collaboration.

3.3.5 Steam Generator Component Models

The current core steam generator component model transfers heat from the intermediate heat transfer loop to water for the generation of steam. Its principal function is to provide this steam generation for the production of electrical power. The interfacial points for the steam generator are the ITHS and the power conversion system models. The source code for all steam generator models for the ALMR and FHR designs is included in the GitHub private repository; it is available for modeling collaboration.

3.3.6 Power Conversion System Models

The current core power conversion system model includes all remaining balance of plant components, including the turbines necessary to generate the power to the grid. Two power conversion system models currently exist, one for the ALMR and one for an FHR. All source code for these models is included in the GitHub private repository; it is available for collaboration.

Feedwater heater component models

The current core feedwater heater component model is a subsystem of the power conversion system model. A feedwater heater is used to preheat water delivered to steam generators. This preheating serves to reduce the irreversibilities and improve the thermodynamic efficiency of the system while helping to avoid thermal shock to the steam generator when the feedwater is introduced back into the steam cycle. A feedwater heater allows the feedwater to be brought up to the saturation temperature very gradually. The heat used to preheat the water is usually derived from steam extracted between the stages of the steam turbine. The percentage of the total cycle steam mass flow used for the feedwater heater is carefully determined to optimize maximum power plant thermal efficiency. Two feedwater heater models have been developed, representing an open and closed feedwater heater, respectively. The source codes for these models are included in the GitHub private repository; they are available for collaboration.

3.3.7 Grid Models

The grid is the end state system model. It provides the interface between power generation and supply distribution outside of the plant and the plant's power production. In the ALMR and FHR models, the electrical generator is modeled the same way as an ideal synchronous generator where the frequency in the electrical connector is the electromotive force of the generator. The frequency of generated electrical current is defaulted to 60 Hz, which can also be adjusted by the user. In practice, the generator frequency must match the grid frequency; otherwise, it can cause a generator trip and disconnection from the grid, which will, in turn, initiate a turbine trip. The source codes for the grid models for the ALMR and FHR designs are included in the GitHub private repository; they are available for modeling collaboration.

3.3.8 Instrument and Controls and Event Driver Models

Individual control system models are populated with distinct control features to provide desired control actions. Two control strategies have been implemented for the ALMR and FHR system models. In Control Strategy #1, PHTS is configured to operate in a load-following scheme with two single-input–single-output (SISO) control loops. As the power generation load changes, the power conversion system adjusts the consumption of steam, which changes the energy flow to the PHTS system. The first PHTS control loop of Control Strategy #1 controls the primary cooling pump speed to maintain a desired temperature setpoint on the PHTS heat exchanger output. The second PHTS control loop of Control Strategy #1 adjusts the reactor control rod position to control reactivity in order to maintain a desired reactor output temperature setpoint. Classical proportional plus integral control methods are used for both control loops.

The IHTS control system model uses a similar approach to the first PHTS control loop. In the case of the IHTS model, the SISO control loop adjusts the intermediate cooling pump speed to maintain the desired IHTS steam generator input temperature. The steam generator will react to load changes, which results in changes in the IHTS steam generator inlet temperature.

The source codes for the I&C models for the ALMR and FHR designs are included in the GitHub private repository; they are available for modeling collaboration.

The introduction of transients into the model simulation can be accommodated through an architectural element known as the “Event Driver.” Details of the ED module (Fig. 29) mirror the overall architecture (as events can occur in any subsystem); the ED module is included in the overall plant model architecture in a similar fashion to the I&C Control modules as seen in Fig. 1. This element interacts with the rest of the end-to-end plant model through the Control Bus. This allows the systematic introduction of initiating events into a simulation.

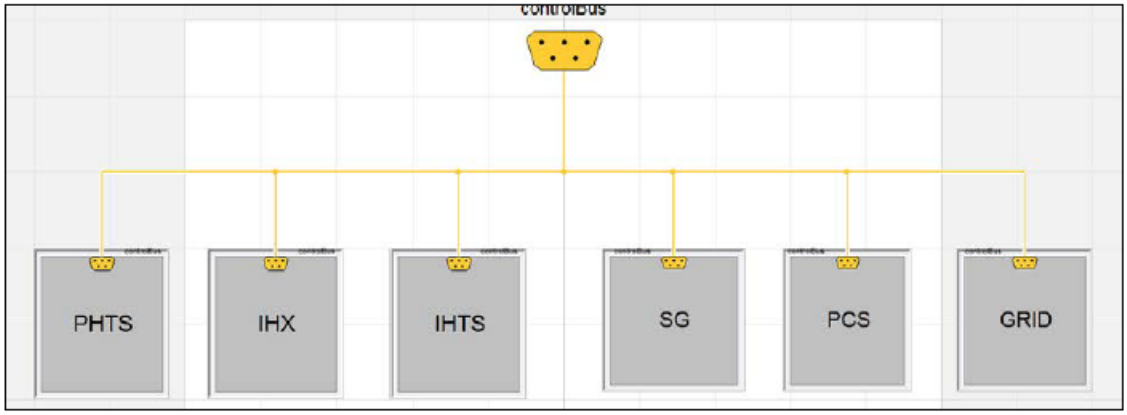


Fig. 29. Event driver module.

3.4 COLLABORATION WORKFLOW

GitHub facilitates collaboration between ORNL and partners by providing access to the model source code for ongoing development. Access to the GitHub repository is controlled by ORNL with authentication. The typical collaboration workflow would focus on the implementation and verification of component or system models by ORNL collaborators. To support model verification, the GitHub repository can also contain a number of component and subsystem software test modules with defined inputs to assist in development and verification by ORNL collaborators. Verified component or system

models can then be integrated into larger systems with higher confidence. System models can then be deployed by ORNL via the web-based application and for offline simulators with FMI Add-in for Excel.

The overall collaboration workflow is as follows:

- ORNL collaborators request and attain access to the GitHub repository. With a shared repository model, a typical approach would involve working on a branch to isolate changes from the main line of development.
- Collaborators work locally to develop and verify new models.
- Changes can be committed and pushed to the branch.
- When the collaborator feels that models are ready to be integrated, a pull request is submitted to the ORNL maintainers to notify of work for review.
- Following a review of the work, ORNL maintainers can merge the branch into the main line of development to integrate the new features.

Software test modules can be supplied as part of the ORNL model repository. While these test rigs are under development, an example workflow is illustrated below for the development of a simplified electrical system.

3.4.1 Example Subsystem Model Test Rig (Electrical)

This section describes a sample workflow for the development of a simplified electrical system component for the PRISM system model architecture. For this use case, the collaborator will develop a new single shaft generator model with a clutch based on the existing implementation PRISM_G1.

A Development package is available in the ORNL repository, with subpackages as shown below. New components are placed in the Components package. TestRigs contains available test modules for testing component and system models. The Tests folder contains executable tests for new components or subsystems.

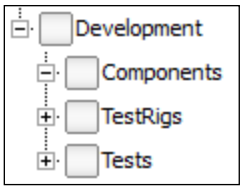


Fig. 30. Example test rig development package.

A component test module exists for the simplified generator model, as represented in Fig. 30. The test software includes a replaceable model for the component to be tested and the appropriate boundary condition specifications.

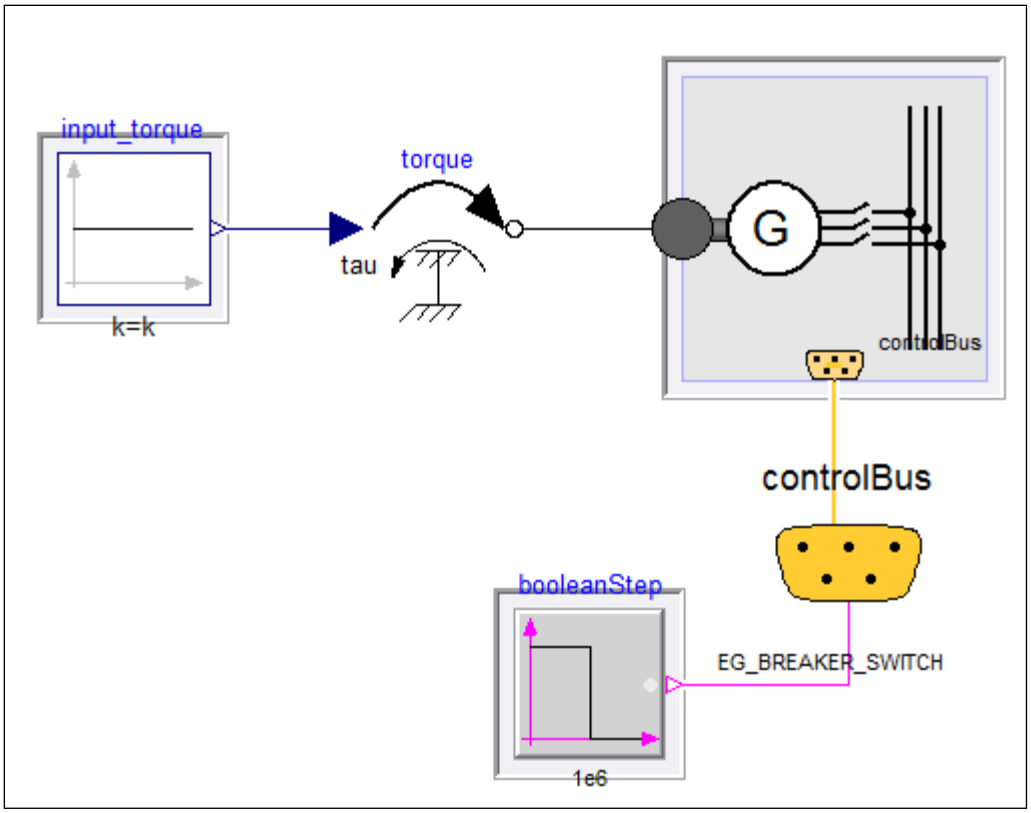


Fig. 31. Electrical component generator model test rig.

The first step in the workflow is to develop a new component model. The model must satisfy the interface requirements between the component and the test software. Component development can extend the interface (i.e., Interfaces.SingleShaftGenerator) or duplicate an existing implementation. In this example case, the existing implementation was duplicated, as shown in Fig. 31, and the new model was added to the Components package.

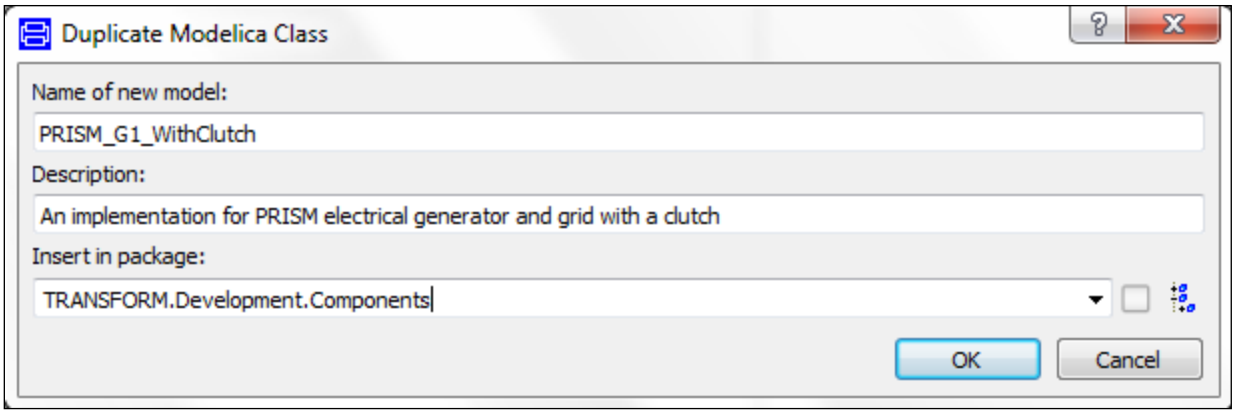


Fig. 32. Dymola dialogue for duplicating model.

The next step is to implement the new model. In this case, the implementation consists of adding a clutch to the model. A new signal is required from the control bus to support this new implementation (and thus would also need to be provided by the control system in the full system model).

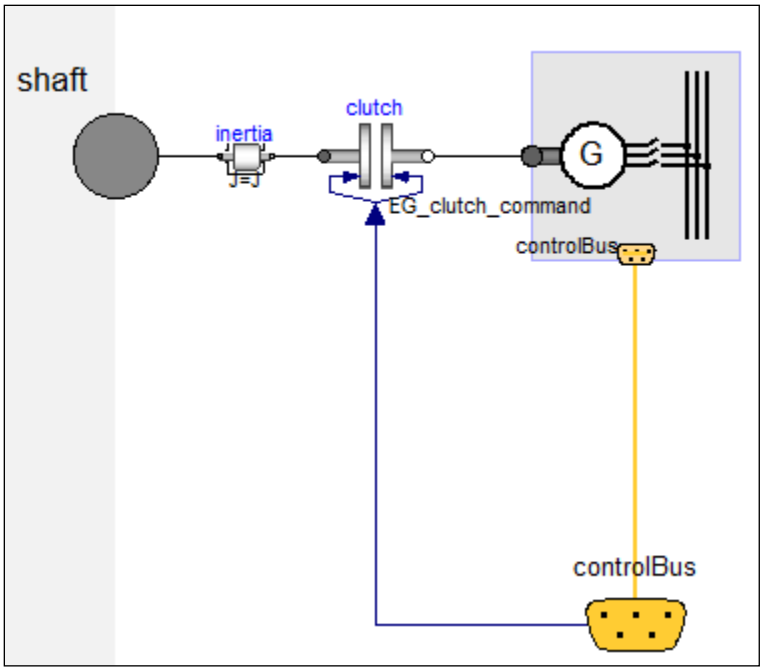


Fig. 33. Dymola implementation of new model with clutch.

After component model implementation is complete, testing can begin. To create a test model, the collaborator extends from the appropriate test rig, as shown in Fig. 33.

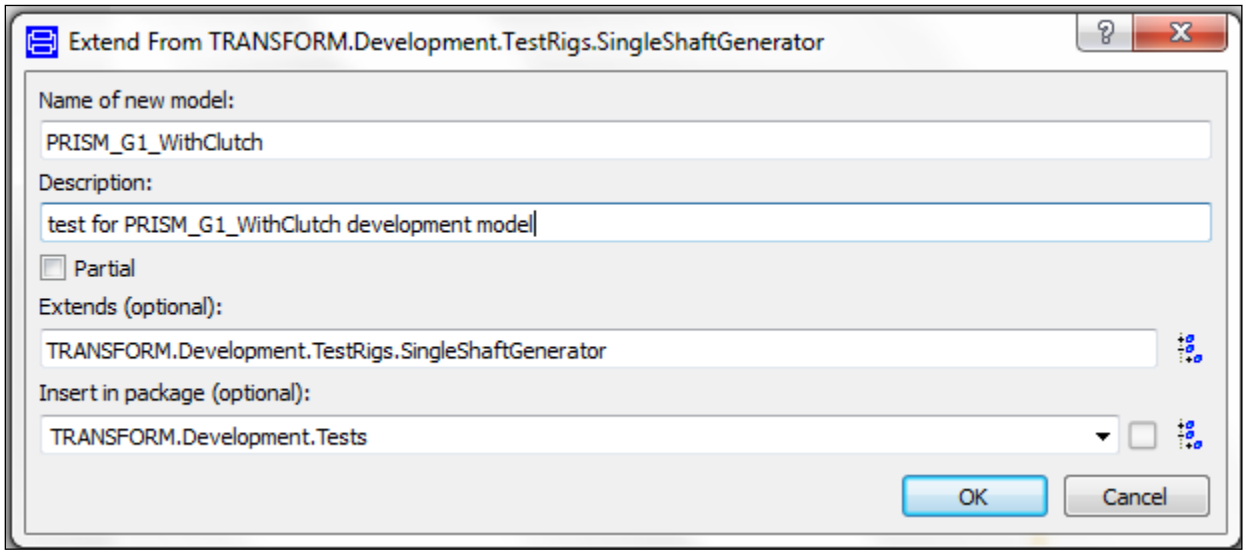


Fig. 34. Dymola dialogue for extending model class.

The new component model should be inserted into the test model by right-clicking on the component, choosing “Change Class,” and then “All Matching Choices,” as shown in Fig. 34. The newly implemented component is an available selection with the description “An implementation for PRISM electrical generator and grid with a clutch.”

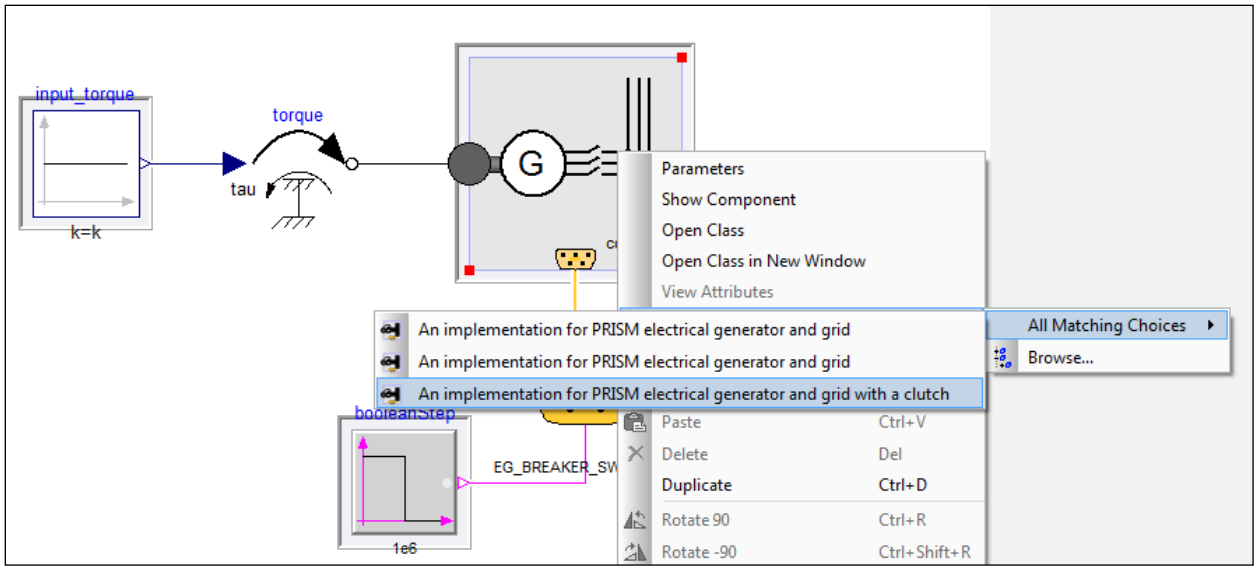


Fig. 35. Dymola model inserted into test rig.

Any new signals required from the control bus must be added to the test model as shown in Fig. 35 for the EG_clutch_command signal. The test model is now complete and ready for testing. Boundary conditions for testing must be provided, and the model can be verified under as many different operating conditions as needed to ensure the robustness and quality of the new component implementation. ORNL developers may also provide specific test conditions that must be simulated and certain behavioral characteristics to be noted in the response to help collaborators with the verification task.

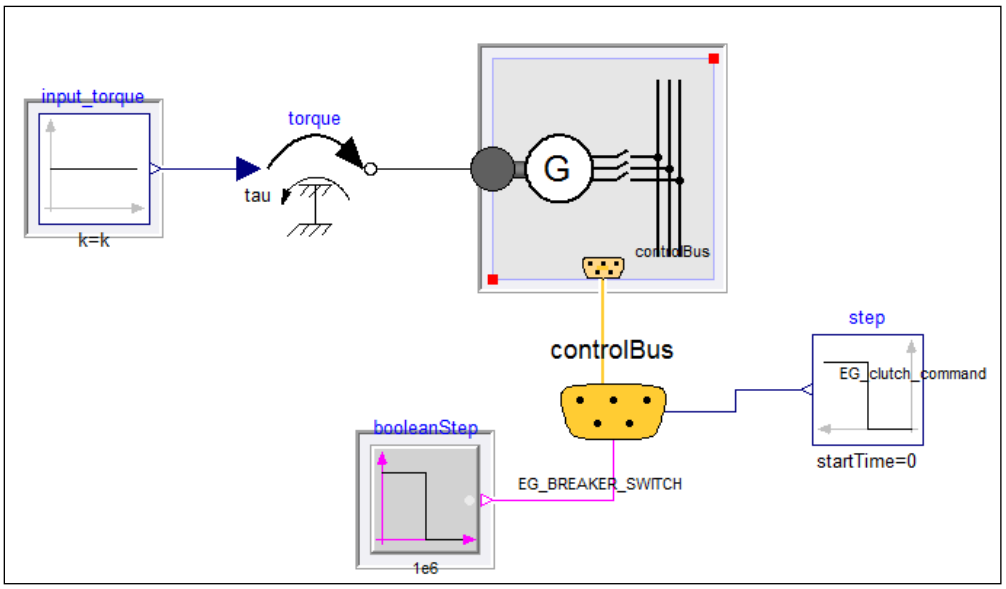


Fig. 36. Signal addition to model test rig.

Once the behavior of the component has been verified, the model can be committed and pushed to the repository, typically on a branch, with a pnuull request sent to an ORNL maintainer to start the discussion for a merge of the new model. After the model is merged into the repository (typically by the ORNL developers), the model can be used in full system models and potentially deployed as a new subsystem option in the web-based application.

REFERENCES

1. *SMR Dynamic System Modeling Tool Update*, ORNL/TM-2013/426, Oak Ridge National Laboratory, Oak Ridge, Tenn., September 2013.

2. R. E. Hale et al., *Update on Small Modular Reactors Dynamic System Modeling Tool*, ORNL/TM-2014/50, Oak Ridge National Laboratory, Oak Ridge, Tenn., March 2014.

3. R. E. Hale et al., *Update on Small Modular Reactors Dynamic System Modeling Tool—Molten Salt Cooled Architecture*, ORNL/TM-2014/322, Oak Ridge National Laboratory, Oak Ridge, Tenn., August 2014.

4. Solomon Hykes, "What is Docker?" *Docker*. https://www.docker.com/whatisdocker/

5. *Prism Preliminary Safety Information Document*, GEFR-00793, UC-87TA, U.S. Department of Energy, Washington, D.C., December 1987.

6. S. R. Greene et al., *Preconceptual Design of a Fluoride-Salt-Cooled Small Modular Advanced High-Temperature Reactor (SmAHTR)*, ORNL/TM-2010/199, Oak Ridge National Laboratory, Oak Ridge, Tenn., 2011.

APPENDIX A. CREATING/UPDATING WEB-BASED APPLICATION

APPENDIX A. CREATING/UPDATING WEB-BASED APPLICATION

As noted in Sect. 2.2.4, updating the web-based application is necessary as additional FMUs become available. In the beta version, updating the web-based application has been simplified to only a few steps. These are described in additional detail below.

INITIALIZATION/UPDATED WEB-BASED APPLICATION

This project was created using the simple command:

```
$ xengen init.
```

This created the `suite.xen` file and the `appdata` directory. The `suite.xen` file was then updated from the default:

```
suite SampleApplication "SampleApplication Application Suite" {}
```

...to...

```
suite ORNL "Oak Ridge Web Application Suite" {}
```

This created the application suite. Then we needed to create a specific application. This was done by running the command:

```
$ xengen newapp --name Fixed --type fmu
```

...inside the `SampleApplication` folder. This creates a directory named `Fixed` and a file inside that directory named `app.xen`. The initial contents of `Fixed/app.xen` were:

```
fmu Fixed "Application Description" {
  description="Put extended description here" "Extended application
description";
  image="images/placeholder.gif";
  labels={} "Maps fmu signal -> nicer name";
  groupByDescription "Use variable description to establish groups";
}
```

Customizing this results in the following:

```
fmu Fixed "A Sample Application" {
  description="A sample application created by @jbatteh";
  image="images/placeholder.gif";
  labels={} "Maps fmu signal -> nicer name";
  groupBySubsystem;
}
```

At this point, the user can actually preview our application with the command:

```
$ xengen serve -w
```

This command instructs `xengen` to start a server on the local machine to host a web-based application locally. The `-w` option instructs `xengen` to monitor the application source for changes and regenerate the application whenever a change is detected.

At this point, the user can point the browser to `http://localhost:8080` and see the web-based application.

ADDING FMUs

Models are added by copying the FMUs into the `Fixed/FMUs` directory.

Now when the application is served again with:

```
xengen serve -w
```

The application can be seen and selected from the available FMUs. The data associated with those FMUs can also be viewed.

COLLABORATION WORKFLOW DETAILS

Collaboration includes the development of modules that can be included in the library of choices for the various reactor system architectures. Before these models can be utilized in end-to-end systems and made available in the web-based application, the modules must be annotated as described in Sect. 2.2.3. The following list and figures detail an overview of the new process for model annotation to support web app creation via FMUs using the simple reactor demo.

- XenGen package provides records for annotating models.
- SimpleReactor demo illustrates usage and resulting FMUs have been used to support web app creation.

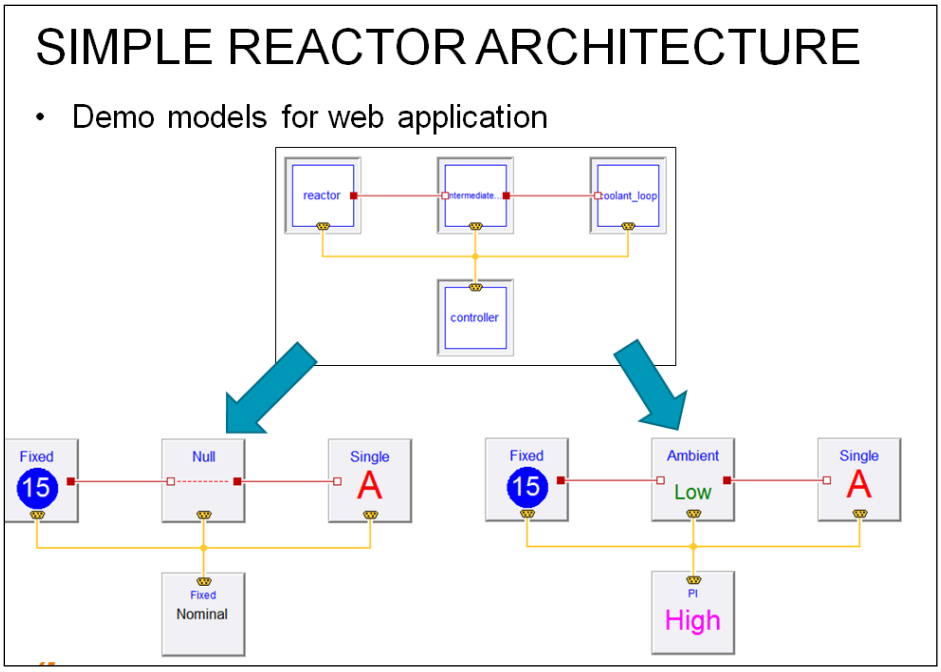


Fig. A.1. Reactor architecture demo.

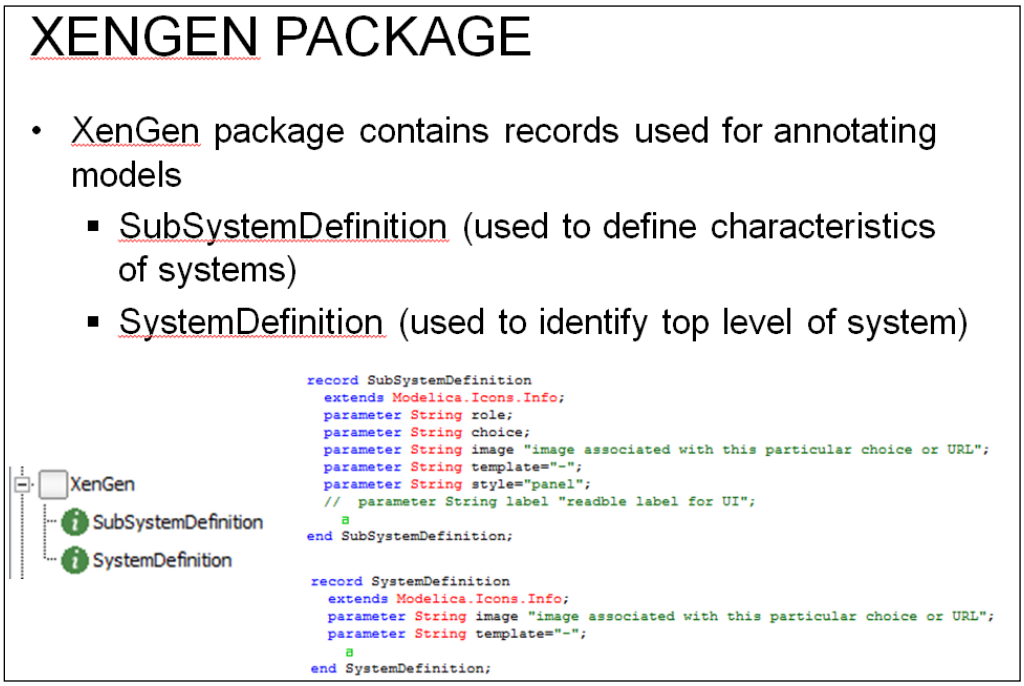


Fig. A.2. Model annotation with Xengen package.

SYSTEMDEFINITION

- Instantiate SystemDefinition at top level of architecture
- Double click on component and set image to be used for architecture

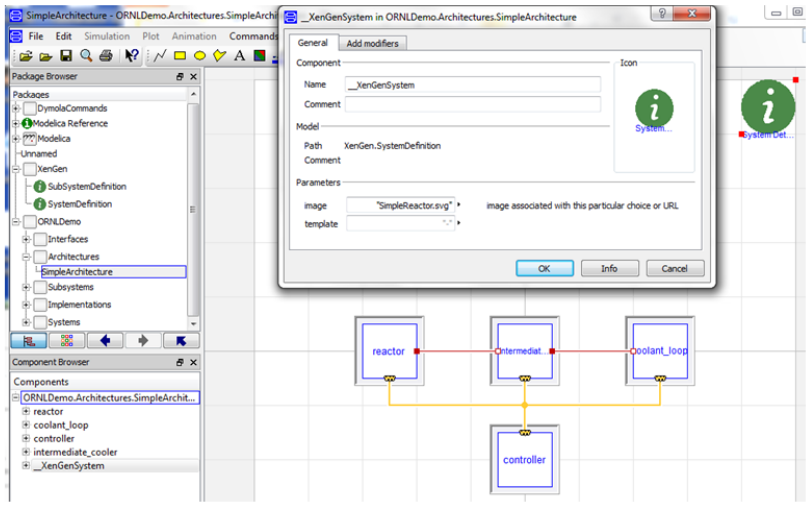


Fig. A.3. System definition steps.

SUBSYSTEM DEFINITION

- Instantiate SubSystemDefinition at top level of each subsystem
  - "role" typically defined in subsystem base class
  - "choice" and "image" typically defined in actual implementations

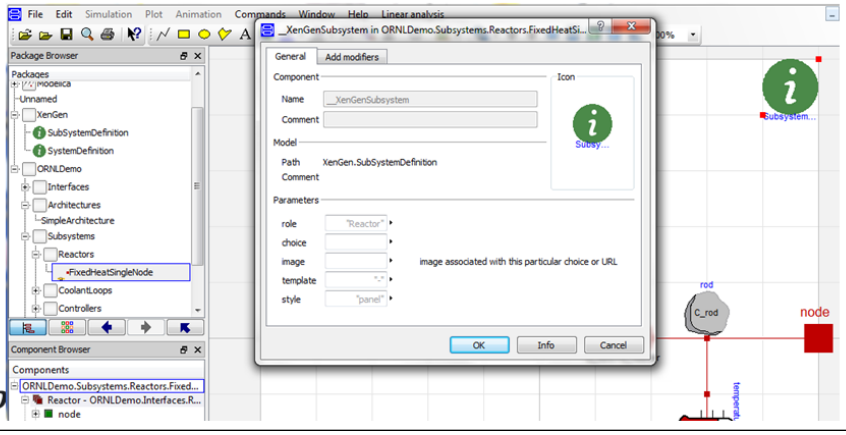


Fig. A.4. Subsystem definition steps.

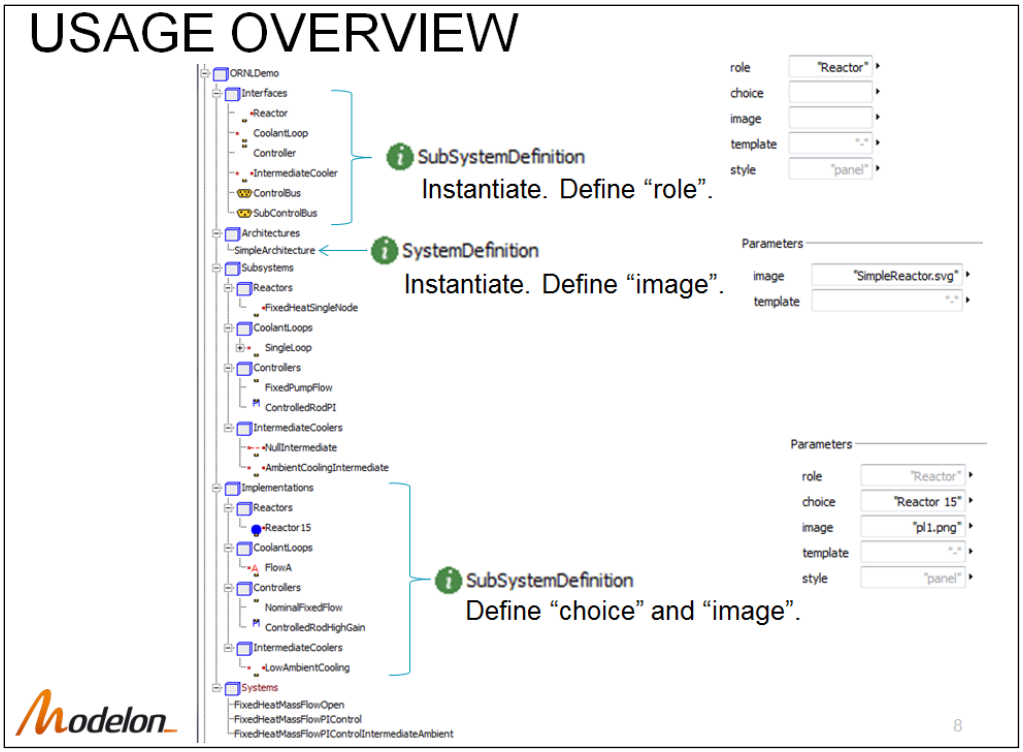


Fig. A.5. Annotation heirarchy.

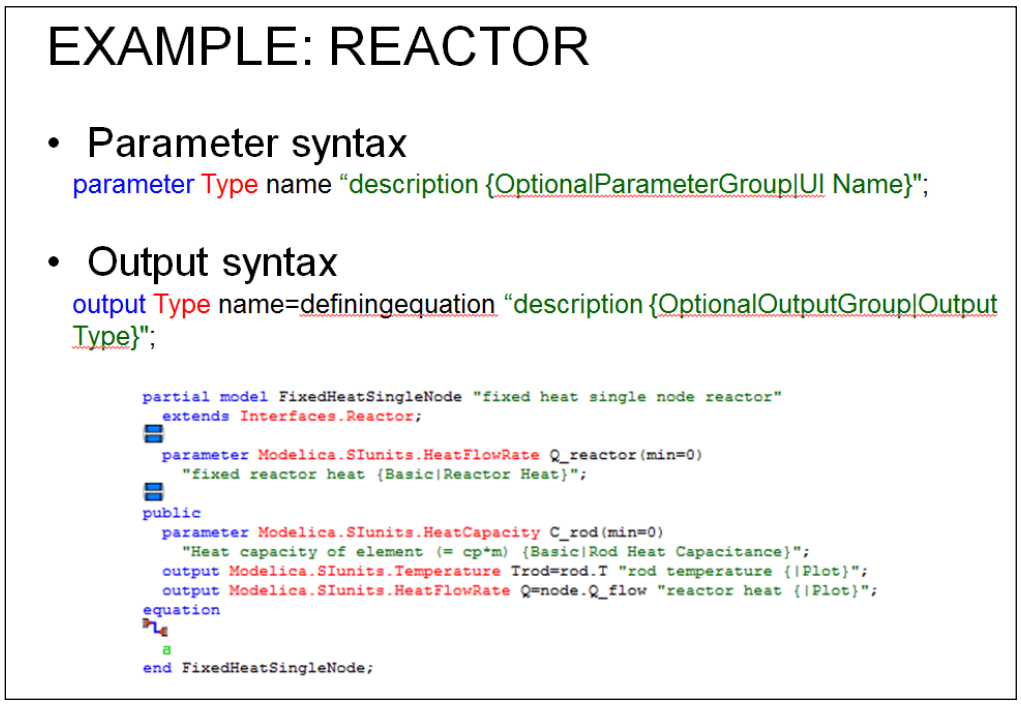


Fig. A.6. Annotation syntax.

- Next steps
  - ORNL commits Modelica source code to GitHub.
  - Modelon annotates existing models based on the older YAML file technique and for any new models to be included in web-based application.

Modelon creates FMUs for web-based applications building by Xo.