

Update on Small Modular Reactors Dynamic System Modeling Tool – Molten-Salt-Cooled Architecture



Approved for public release;
distribution is unlimited.

Richard Hale
Sacit Cetiner
David Fugate
A.Louis Qualls
Ethan Chaleff
Doug Rogerson
Robert Borum
John Batteh
Michael Tiller

August 2014

DOCUMENT AVAILABILITY

Reports produced after January 1, 1996, are generally available free via US Department of Energy (DOE) SciTech Connect.

Website <http://www.osti.gov/scitech/>

Reports produced before January 1, 1996, may be purchased by members of the public from the following source:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone 703-605-6000 (1-800-553-6847)
TDD 703-487-4639
Fax 703-605-6900
E-mail info@ntis.gov
Website <http://www.ntis.gov/help/ordermethods.aspx>

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange representatives, and International Nuclear Information System representatives from the following source:

Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831
Telephone 865-576-8401
Fax 865-576-5728
E-mail reports@osti.gov
Website <http://www.osti.gov/contact.html>

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.

Reactor and Nuclear Systems Division

**UPDATE ON SMALL MODULAR REACTORS DYNAMIC SYSTEM MODELING TOOL –
MOLTEN-SALT-COOLED ARCHITECTURE**

Richard Hale
Sacit Cetiner
David Fugate
A. Louis Qualls
Ethan Chaleff
Doug Rogerson
Robert Borum
John Batteh
Michael M. Tiller

Date Published: August 2014

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
LIST OF TABLES	ix
ACRONYMS	xi
ACKNOWLEDGMENTS	xiii
EXECUTIVE SUMMARY	xv
1. INTRODUCTION	1
1.1 BACKGROUND	1
1.2 REPORT ORGANIZATION	1
2. MODEL DEVELOPMENT	1
2.1 TASK DESCRIPTION	1
2.2 RVACS/DRACS	2
2.2.1 RVACS	2
2.2.2 DRACS	3
2.3 FEEDWATER HEATER	5
3. MOLTEN-SALT-COOLED ARCHITECTURE	7
3.1 TASK DESCRIPTION	7
3.1.1 Fluoride High-Temperature Reactors	8
3.1.2 SmAHTR Reactor Concept	10
3.2 MODELICA MODEL FOR SMAHTR	15
3.2.1 Passive Safety System	15
3.2.2 Reactor/Primary Heat Transport System	15
3.3 PRIMARY HEAT TRANSPORT SYSTEM	34
3.4 INTERMEDIATE HEAT EXCHANGER	37
3.5 INTERMEDIATE HEAT TRANSPORT SYSTEM	38
3.6 STEAM GENERATOR MODEL	39
3.7 POWER CONVERSION SYSTEM	41
3.7.1 Null Implementation	41
3.7.2 Steam Separator Model	42
3.8 ELECTRICAL GRID	43
4. MOLTEN-SALT-COOLED I&C IMPLEMENTATIONS	43
4.1 TASK DESCRIPTION	43
4.2 ADDITIONAL I&C CONTROL IMPLEMENTATIONS	46
4.2.1 Steam Generator Control Implementations	46
4.2.2 Main Steam Turbine Control Implementation	47
5. COLLABORATION IMPLEMENTATION	51
5.1 TASK DESCRIPTION	51
5.2 MODEL REPOSITORIES	52
5.3 COLLABORATION PARTNERS/PROTOCOL	52
5.3.1 Collaboration Partners	52
5.3.2 Web-Based Collaboration	53
6. INTEGRATING ACTIVITIES	54
6.1 REMAINING FY 2014 TASKS	54
6.2 ADVANCED REACTORS I&C INTEGRATION ACTIVITIES	54
6.2.1 Supervisory Control	54
6.2.2 Advanced PRA	55
6.2.3 Hybrid Energy Systems	55

7. REFERENCES	57
APPENDIX A. INTRODUCTORY COLLABORATION MATERIAL	A-1
Collaboration Protocols	A-10
Getting Started	A-10

LIST OF FIGURES

Figure	Page
Fig. 1. RVACS component model.	3
Fig. 2. DRACS component model.	4
Fig. 3. Dynamic response of DRACS system.	5
Fig. 4. Closed feedwater heater component model.	6
Fig. 5. Open feedwater component model.	7
Fig. 6. Fluoride high-temperature reactor.	9
Fig. 7. Top-level architecture of the SmAHTR dynamic simulation tool for electricity generation.	9
Fig. 8. Top-level architecture of the SmAHTR dynamic simulation tool for hydrogen production.	10
Fig. 9. SmAHTR reactor vessel (isometric).	11
Fig. 10. SmAHTR reactor vessel (cross-section).	12
Fig. 11. SmAHTR design parameters.	13
Fig. 12. SmAHTR design parameters.	13
Fig. 13. SmAHTR design parameters.	14
Fig. 14. SmAHTR integral systems configuration.	14
Fig. 15. Reactor kinetics model object with inputs and outputs.	15
Fig. 16. User dialog for specifying geometry and neutronic parameters for the ReactorKinetics class.	16
Fig. 17. Thermal conductivity of nuclear-grade graphite as a function of temperature at different fluence levels.	18
Fig. 18. Variation of specific heat capacity as a function of temperature for grade H-451 graphite.	19
Fig. 19. Thermal conductivity of Hastelloy-N as a function of temperature [HAYNES –Reference[4]].	21
Fig. 20. Specific heat capacity of Hastelloy-N as a function of temperature [HAYNES-Reference[4]].	22
Fig. 21. Power density profile as a function of axial position.	25
Fig. 22. Thermal conduction model block for SmAHTR fuel.	25
Fig. 23. Dimensions for the fuel thermal conduction model (adiabatic boundary represents the symmetry condition).	26
Fig. 24. Geometry of a fuel pin for thermal model (quantities in mm).	28
Fig. 25. Fuel and coolant axial temperature profiles for fully irradiated fuel.	29
Fig. 26. Lateral fuel temperature profile at midplane for fully irradiated fuel.	30
Fig. 27. Fuel and coolant axial temperature profiles for unirradiated fuel.	31
Fig. 28. Lateral fuel temperature profile at midplane for unirradiated fuel.	32
Fig. 29. Geometry of a SmAHTR assembly.	32
Fig. 30. A single flow channel.	33
Fig. 31. Nodal mean coolant temperature profile along the channel.	33
Fig. 32. Nodal heat transfer coefficient profile along the channel.	34
Fig. 33. Single-zone representation of the reactor core and the connected fluid systems.	36
Fig. 34. Two-zone representation of the reactor core and connected fluid systems.	36
Fig. 35. SmAHTR IHX model.	37
Fig. 36. Temperature profiles on the shell and the tube side as function of axial node.	38
Fig. 37. Intermediate heat transport system.	39
Fig. 38. Implementation of steam generator for FHR.	40
Fig. 39. Effect of plenums on steam generator dynamics.	40
Fig. 40. PCS Null implementation.	41
Fig. 41. FHR power conversion system with steam separation.	42

Fig. 42. Steam separator component.....	43
Fig. 43. Reactor I&C control strategy #1 (ALMR transient).....	44
Fig. 44. Reactor I&C control strategy #2 (ALMR transient).....	45
Fig. 45. Architecture selection I&C control strategy #1 (ALMR model).....	46
Fig. 46. Steam generator functional diagram.....	47
Fig. 47. Turbine speed control concept diagram.....	47
Fig. 48. Baseline model PCS with throttle valve added to the high-pressure turbine supply.....	48
Fig. 49. Throttle valve parameters.....	49
Fig. 50. Power plant dynamic test shifting power from 111 MW to 151 MW with high-pressure steam turbine throttle valve control.....	50
Fig. 51. Power plant dynamic test shifting power from 111 MW to 151 MW with high-pressure steam turbine throttle valve control.....	51
Fig. 52. Architecture for local supervisory control system and functional modules of decision-making.....	56
Fig. A.1. ORNL SMR background presentation – Slide 1.....	A-3
Fig. A.2. ORNL SMR background presentation – Slide 2.....	A-4
Fig. A.3. ORNL SMR background presentation–Slide 3.....	A-4
Fig. A.4. ORNL SMR background presentation–Slide 4.....	A-5
Fig. A.5. ORNL SMR background presentation–Slide 5.....	A-5
Fig. A.6. ORNL SMR background presentation–Slide 6.....	A-6
Fig. A.7. ORNL SMR background presentation–Slide 7.....	A-6
Fig. A.8. ORNL SMR background presentation–Slide 8.....	A-7
Fig. A.9. ORNL SMR background presentation–Slide 9.....	A-7
Fig. A.10. ORNL SMR background presentation–Slide 10.....	A-8
Fig. A.11. ORNL SMR background presentation–Slide 11.....	A-8
Fig. A.12. ORNL SMR background presentation–Slide 12.....	A-9
Fig. A.13. ORNL SMR background presentation–Slide 13.....	A-9
Fig. A.14. ORNL SMR modeling web site.....	A-11
Fig. A.15. ORNL SMR modeling brochure.....	A-13
Fig. A.16. TortoiseGit download.....	A-15
Fig. A.17. ORNL GitHub SMR modeling repository.....	A-18
Fig. A.18. Local machine Git/GitHub icons.....	A-19
Fig. A.19. Local machine repositories (application view).....	A-19
Fig. A.20. GitHub repositories (browser view).....	A-20
Fig. A.21. Local machine GitHub folder.....	A-20
Fig. A.22. Windows Git options (right click on folder).....	A-21
Fig. A.23. Local repositories viewed through GitShell.....	A-21
Fig. A.24. TortoiseGit download Step #1.....	A-22
Fig. A.25. TortoiseGit download Step #2.....	A-23
Fig. A.26. TortoiseGit download Step #3.....	A-23
Fig. A.27. TortoiseGit download Step #4.....	A-24
Fig. A.28. TortoiseGit download Step #5.....	A-24
Fig. A.29. TortoiseGit download Step #6.....	A-25
Fig. A.30. TortoiseGit download Step #7.....	A-25
Fig. A.31. TortoiseGit download Step #8.....	A-26
Fig. A.32. TortoiseGit download Step #9.....	A-26
Fig. A.33. TortoiseGit tutorial–Create Repository.....	A-27
Fig. A.34. TortoiseGit tutorial–Add Codebase to Repository.....	A-27
Fig. A.35. TortoiseGit tutorial–Change File and Commit.....	A-28
Fig. A.36. TortoiseGit tutorial–Clone the Repository.....	A-28
Fig. A.37. “git status” command for determining current status of files in directory.....	A-29

Fig. A.38. “git add” command for initiating tracking of changes in files.....	A-29
Fig. A.39. “git commit -m” command for committing changes in the file to the local repository and providing explanation (i.e., -m “added RVACS_demoTotal.mo to repository”).	A-30
Fig. A.40. Local history and view of changes to file.	A-31
Fig. A.41. Local history and view of changes to file.	A-31
Fig. A.42. Modelica model change in steel density.	A-32
Fig. A.43. Git “commit” of change to model.....	A-32
Fig. A.44. Git “pull” of change to model to update repository.....	A-33
Fig. A.45. TortoiseGit GUI denoting density model changes.....	A-34
Fig. A.46. Dymola model modified (date modified).	A-34
Fig. A.47. Dymola model.	A-35
Fig. A.48. Model modification reflected in Dymola parameter interface.....	A-35

LIST OF TABLES

Table	Page
Table 1. Selected thermo-physical properties of grade H-451 graphite.....	17
Table 2. Thermal properties of Hastelloy-N	20
Table 3. Flibe thermal hydraulic parameters	34
Table 4. Flinak thermal-hydraulic parameters	35
Table 5. KF-ZrF ₄ thermal-hydraulic parameters.....	35

ACRONYMS

AHTR	advanced high-temperature reactor
ALMR	advanced liquid-metal reactor
BDF	backward differentiation formulae
BOL	beginning of life
DAE	differential/algebraic equation
DAR	DRACS air radiator
DOE	US Department of Energy
DRACS	direct reactor auxiliary cooling system
emf	electromotive force
EOL	end of life
FHR	fluoride high-temperature reactor
HTGR	high-temperature gas-cooled reactor
I&C	instrumentation and controls
ICHMI	Instrumentation, Control, and Human-Machine Interface
ICL	intermediate cooling loop
IHX	intermediate heat exchanger
ILP	intermediate loop pump
LWR	light water reactor
MCL	main circulating loop
MCP	main circulating pump
MSR	molten salt reactor
NGNP	next-generation nuclear plant
ODE	ordinary differential equation
ORNL	Oak Ridge National Laboratory
PCS	power conversion system

PHX	primary heat exchanger
PRISM	power reactor innovative small module
QA	quality assurance
RSC	reserve shutdown control
RVACS	reactor vessel auxiliary cooling system
SmAHTR	small modular advanced high-temperature reactor
SMR	small modular reactor
STP	standard temperature and pressure
TRISO	tri-structural isotropic
V&V	verification and validation

ACKNOWLEDGMENTS

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

The authors wish to thank the technical peer reviewers and editorial staff at Oak Ridge National Laboratory for their feedback and assistance in improving this report.

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 multiple coupled reactors at a single site. The objective of the project is to provide a common simulation environment and baseline modeling resources to facilitate rapid development of dynamic advanced reactor SMR models, ensure consistency among research products within the Instrumentation, Controls, and Human-Machine Interface (ICHMI) 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 (MoDSIM) 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.

To date, we now have the ability to rapidly configure multiple modules together into various integrated end-to-end reactor systems for two reactor architectures: (1) advanced liquid-metal reactor (ALMR) and (2) a fluoride-salt-cooled high temperature reactor (FHR). Additionally, the ability to perform simple transients (reactivity transient for this demonstration) on these models and evaluate the results has also been demonstrated. Finally, the ability to modify the reactor configuration and produce these results “on-the-fly” in Excel locally as well as through a web application has been demonstrated. The work performed to date represents a significant advance in modeling capabilities that can be leveraged across the Advanced Reactor Technology R&D Program.

Since the update of March 2014, the following work has been completed or is under development:

- 1) The addition of a second advanced reactor architecture (FHR) has been included in the model library.
- 2) The addition of a passive safety system, reactor vessel auxiliary cooling system/direct reactor auxiliary cooling system (RVACS/DRACS), has been included in the architecture.
- 3) Another feedwater heater balance-of-plant module has been added.
- 4) The previous instrumentation and control (I&C) reactor control implementations [temperature and temperature differential (dT) control] are being incorporated into the new architecture.
- 5) Additional I&C models are under development for both a steam generator and main steam turbine.
- 6) Additional material has been included in the user’s manual (associated with collaboration work flows).
- 7) A web site (<http://smrdev.ornl.gov/>) and a web-based Oak Ridge National Laboratory (ORNL) model repository are up and running. Initial collaborations have been established with the following partners:
 - a. Idaho National Laboratory
 - b. New Mexico State University
 - c. North Carolina State University
 - d. Massachusetts Institute of Technology

- e. Georgia Institute of Technology
- f. Ohio State University

8) The first model (ALMR end-to-end model) was delivered to a related ICHMI project for use (I&C supervisory control project).

The remaining work for FY 2014 includes the delivery of a first release for the web application along with further refinement and establishment of initial collaboration protocols and procedures with our national lab, university and business partners. FY 2015 work is still in the planning stages but is expected to include (1) the establishment of quality assurance requirements for the models, (2) a technology control plan to identify the limits and boundaries for model sharing and collaboration, (3) initiation of first collaboration data/model sharing with our university and government partners, (4) further development of balance-of-plant models, and (5) work towards the extension of the model library to a third advanced reactor concept (initially anticipated to be high-temperature gas-cooled reactor).

1. INTRODUCTION

1.1 BACKGROUND

As noted in ORNL's previous update for the Small Modular Reactor (SMR) Dynamic System Modeling Tool (MoDSIM) in March 2014 [1], the project is intended to develop simulation resources and tools to allow collaborative modeling and control study of various advanced (non-light water reactor [LWR]), small modular reactor (SMR) configurations, including the use of multiple, interconnected reactor units at a single site. 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. Additional details associated with the project's high-level goals and objectives can be found in Reference [1] and will not be reproduced here. This effort is in the third year of development, and this report serves as an update on the progress of this project. The project has been funded for FY 2015 with a final report expected in FY 2015.

1.2 REPORT ORGANIZATION

The report is divided into six sections. Besides the Introduction (Sect. 1), these include Model Development (Sect. 2), Molten-Salt-Cooled Architecture (Sect. 3), Molten-Salt-Cooled I&C Implementation (Sect. 4), Collaboration Implementation (Sect. 5), and Integrating Activities (Sect. 6). In Sect. 2, the model development includes modifications to the existing baseline component models either for incorporation into a molten salt architecture or that serve as upgrades/additions to the existing library. In Sect. 3, the development of a molten salt architecture includes the specific changes necessary to modify the existing advanced liquid-metal reactor (ALMR) architecture to include molten-salt-cooling typified by the fluoride-salt-cooled high-temperature reactor (FHR). Similarly, Sect. 4 includes a discussion of ongoing efforts to implement the I&C architectures developed for the ALMR model into the molten-salt-cooled architecture. Also discussed in this section is the development of other control systems necessary as part of the reactor design, including the steam generator control and the main steam turbine control. This work is under development and expected to be completed for the December 2014 deliverable. Ongoing collaborative work is discussed in Sect. 5, which includes both the list of potential collaboration partners and the development of the initial Oak Ridge National Laboratory (ORNL) model repository and protocols for collaboration. Finally, in Sect. 6, ongoing integration activities with other US Department of Energy (DOE) advanced reactor/SMR projects are discussed.

2. MODEL DEVELOPMENT

2.1 TASK DESCRIPTION

Models are under continual development in this project. The previous status report [1] documented the first complete end-to-end system model. This model was the ALMR power reactor innovative small module (PRISM) design and included the following systems/components:

1. primary heat transport system,
2. intermediate heat exchanger,
3. intermediate heat transport system,
4. steam generator,
5. power conversion system, and
6. grid.

In addition to these components, several additional architectural elements were created, including an I&C architectural element and an Event Driver element. These elements are connected to system/component models through an architectural bus that allows control systems to be overlaid and transients to be introduced into the models. These architectural elements and the connecting bus are retained in the current molten-salt-cooled model architecture described in this report and will be in all future architectures. The molten-salt-cooled architecture is similar to the ALMR architecture. Most of the models can be modified to account for different fluid properties and operational environments. A description of these modifications can be found in Sect. 3. In addition to these modifications, new models have been developed, including the passive safety system DRACS (direct reactor auxiliary cooling system) and RVACS (reactor vessel auxiliary cooling system) and a feedwater heater model for use in an expanded power conversion system (PCS). These models are described in the next two sections.

2.2 RVACS/DRACS

Passive safety systems are considered in both ALMR and FHR concepts. These passive safety systems include what is termed a RVACS or a DRACS. One or both of these are incorporated in advanced reactor designs to serve as the ultimate heat sink. Both of these systems make use of heat exchangers in direct or indirect communication between the primary coolant in the vessel and the external environment. A description of the RVACS and DRACS models is provided in the following section, along with the Modelica implementation of the models.

2.2.1 RVACS

RVACS functions by removing residual heat from the reactor vessel via air-cooled heat exchangers to cool the vessel and, thus, the core. In order to incorporate an RVACS model into the advanced SMR modeling tool, a new component “RVACS” was developed. Rather than calculating the density change and attempting to solve the pressure-driven flow equation, which can result in numerical instabilities, the flow is calculated based on the stack equation. The stack equation is an empirical relation for chimney flow rates with air as the primary working fluid.

$$Q_{stack} = C * A \sqrt{2 g h \frac{T_i - T_o}{T_i}},$$

where C is the discharge coefficient, approximately 0.65 at Standard Temperature and Pressure (STP), A is the cross-sectional area of the chimney, h is the height of the chimney, T_i is the temperature inside the chimney, and T_o is the ambient temperature. Although this method does not take into account some of the faster dynamic effects, the time constants of RVACS are much smaller than those of heat transport through the vessel, and so this difference should be negligible.

Using the flow rate calculated from the stack equation, the model determines the heat transfer coefficient using the Dittus-Boelter relation and tracks the increasing temperature of the air along the vessel, using the outlet temperature to calculate the stack flow, creating the system coupling. At each node, the model solves

$$\rho \delta V c_p \frac{\partial T[j]}{\partial t} = \phi[j] \delta A + \dot{m} c_p (T[j] - T[j - 1]).$$

The Dymola graphical representation of the model shown in Fig. 1 has one connector port, a thermal port, which passes values for heat flux, temperature and heat transfer coefficient. It has no outlet port, as the exhaust is released into the atmosphere and not tracked.

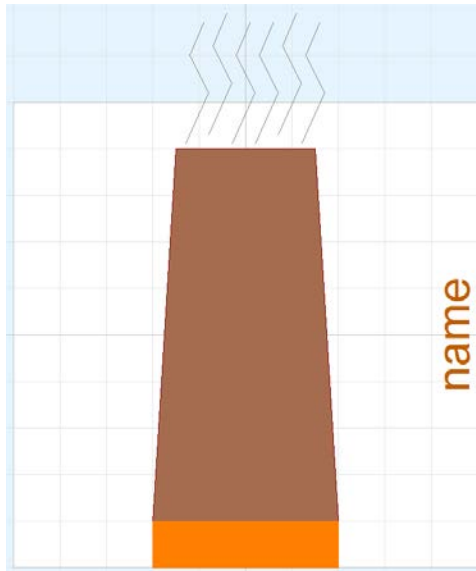


Fig. 1. RVACS component model.

2.2.2 DRACS

DRACS functions based on a principle similar to RVACS by exploiting natural-convection-driven flow to provide a negative feedback to the temperature of the core. As core temperature increases, so does the heat removed by DRACS. 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 located 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. DRACS is implemented into the ORNL ModSim environment by connecting to the downcomer in the primary heat transport system. The DRACS component uses 1-D pipe flow models containing the governing equations for energy and mass transport in the system. The Modelica fluid library accounts for thermal expansion with increased temperature and allows pressure difference across the pipes to drive the flow in the system. An accumulator is necessary to allow expansion of the fixed mass in the system. The DRACS component model is shown in Fig. 2. It uses FLiNaK (in this example) as a working fluid and connects by a thermal port to the external modeling environment. DRACS systems are envisioned to have multiple parallel coolant systems. The DRACS model contains a parameter “n_parallel” that can be used as a simplified approximation for n identical components, or set to 1 to model parallel systems explicitly.

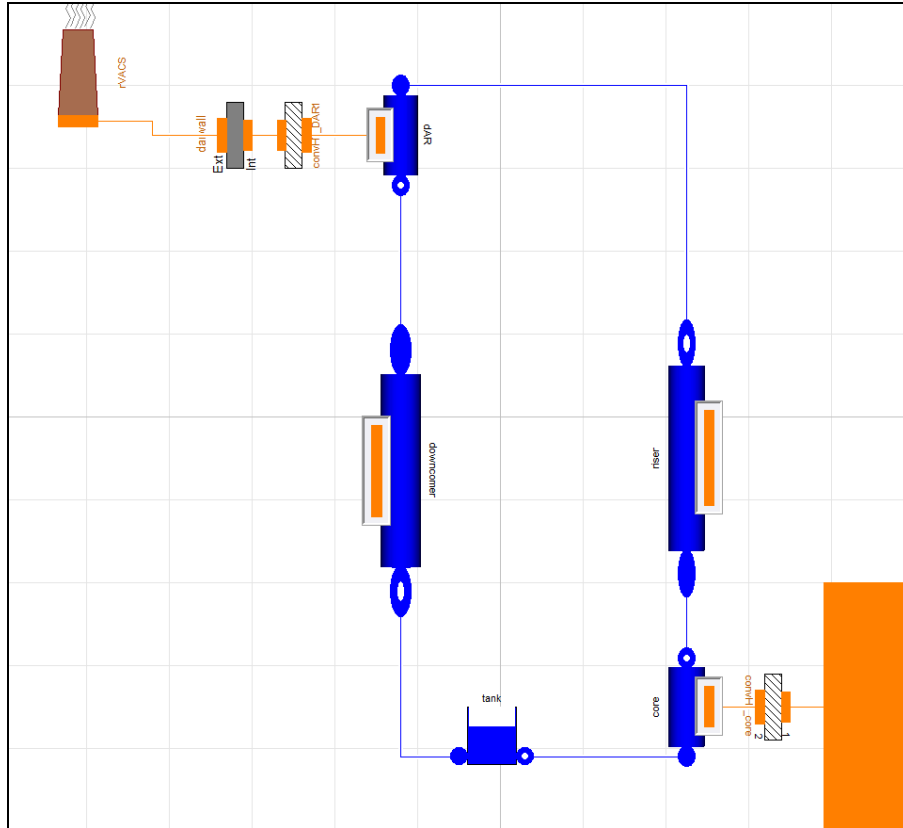


Fig. 2. DRACS component model.

The component values for the system are taken from the preconceptual design document for the small modular advanced high-temperature reactor (SmaHTR) system, but these values are limited. The performance of the system is strongly governed by the friction pressure drop through the core-side heat exchanger and DRACS-air radiator (DAR). Design data are not available for these components, so the values at present are estimates that give reasonable values for performance.

At steady state, the secondary temperature rise in the heat exchanger is 26°C, compared to 30°C in the design documentation for SmaHTR. The flow rate through the system is slightly higher at 13.3 kg/s compared to 9 kg/s, but this is highly dependent on the frictional pressure drops in the system. As a demonstration, the SmaHTR DRACS model was set up with an input of 0.42 MW through three parallel implementations, one-third of the design limit for heat rejection through DRACS. After reaching steady state, the input power was rapidly ramped to 1.26 MW, 100% of the design limit. The response of the system is shown in Fig. 3, where “core.Q” is the heat input to the system in kW, “power_rejected” is the heat lost through the radiator in kW, and “core.T [3]” is the exit temperature of the DHX.

These results are included to demonstrate that a working DRACS model has been implemented. Work is ongoing to confirm the accuracy of numerical results.

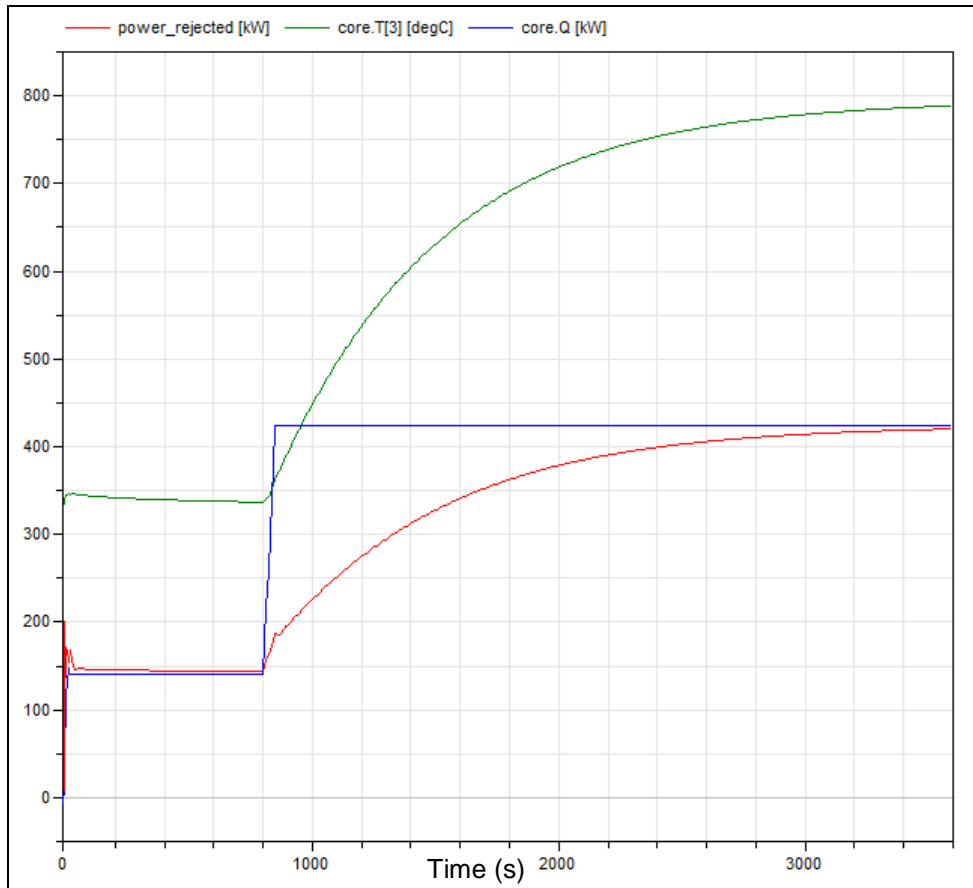


Fig. 3. Dynamic response of DRACS system.

2.3 FEEDWATER HEATER

A feedwater heater is used to pre-heat 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.

There are two different types of feedwater heaters. Open feedwater heaters are essentially mixing volumes, where two flows of liquid combine into one exit stream. These require both streams to be at the same pressure in order to mix. Open feedwater heaters require pumps after each stage as the pressure difference between successive stages must be increased. Closed feedwater heaters are heat exchangers, typically tube and shell, where fluid drawn off by extraction from the steam turbine or condensate from the steam separator passes on one side and feedwater from the condenser is heated before entering the boiler. Since the two fluids do not mix, they can be at different pressures.

The addition of feedwater heaters to the balance-of-plant model is important toward realizing a more realistic PCS.

Modelica Closed Feedwater Heater

The closed feedwater heater model (Fig. 4) is a different implementation of the existing heat exchanger model. All the flow parameters for the shell and the tube can be modified as in the existing heat exchanger architecture.

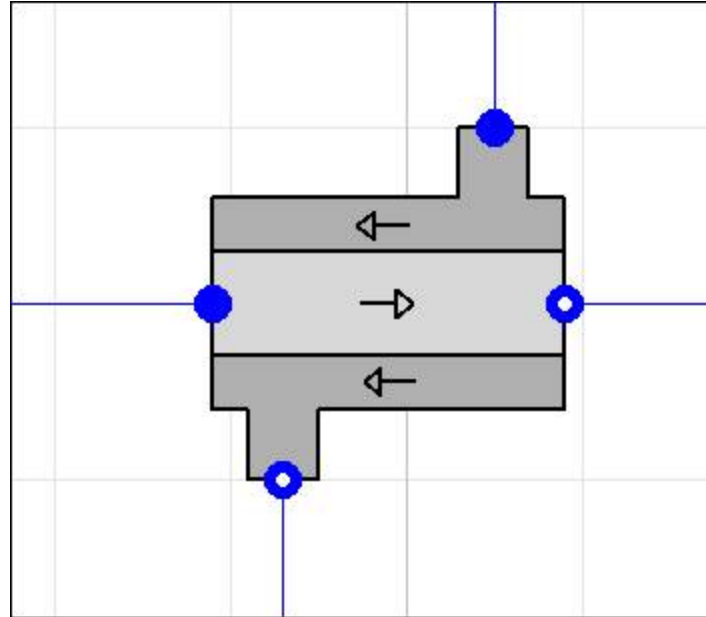


Fig. 4. Closed feedwater heater component model.

Since the feedwater heater might receive low-quality saturated water on the shell side from turbine bleeds, the shell pipe model uses a two-phase correction to the Dittus-Boelter correlation for heat transfer coefficient calculation, as opposed to the standard Dittus-Boelter correlation commonly used to determine the nusselt number:

$$Nu_D = 0.023 Re_D^{0.8} Pr^{0.4} .$$

This correlation is used for subcooled or supersaturated flow in the pipe. For high-quality steam above the critical heat flux, only the mass flow rate of steam is used in the calculation of the nusselt number. For boiling or condensing flow in the pipe, a constant heat transfer coefficient for boiling flow is used. This is acceptable for a feedwater heater, since the model should receive very low-quality or sub-cooled steam, whereas for a turbine bleed heat exchanger, the heater should receive very high-quality steam. If future investigation proves otherwise, a pipe model with the Chen correlation also currently exists and could be used.

Open Feedwater Heater

The open feedwater heater model (Fig. 5) is implemented at present by connecting a mixing volume and an accumulation tank in series.

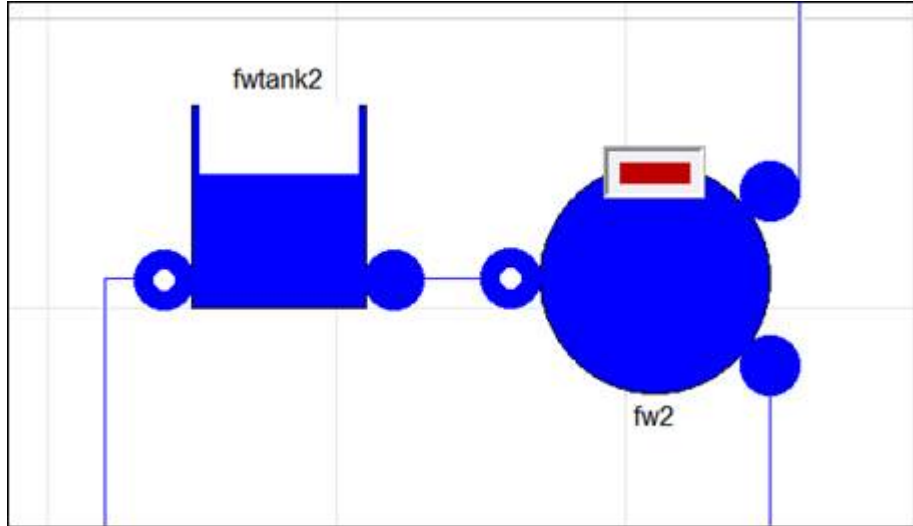


Fig. 5. Open feedwater component model.

The two flows mix in the open feedwater heater, so there is no attempt to determine the heat transfer properties of the flows. Rather, the model tracks energy conservation for the mixing flows:

$$\text{der}(E) = h_{f1}q_{f1} + h_{f2}q_{f2} + h_{out}q_{out} .$$

The other governing equation is defining the change in the mixing volume as zero:

$$\text{der}(V) = \frac{1}{\rho} (q_{f1} + q_{f2} + q_{out}) = 0 .$$

The tank model is used to account for variations in flow rate of the feed to the pump, allowing for an accumulation of mass in the feedwater heater. The outlet pressure of the tank model is

$$p = \rho \times g \times V_{tank}/A .$$

3. MOLTEN-SALT-COOLED ARCHITECTURE

3.1 TASK DESCRIPTION

As noted previously, the goal of the project is to provide a common simulation environment and baseline modeling resource to facilitate rapid development of dynamic SMR models for control strategy development and evaluation. Additional objectives include ensuring consistency among research products within the Instrumentation, Control, and Human-Machine Interface (ICHMI) technical area and leveraging cross-cutting capabilities while minimizing duplication of effort. The development of this capability across the range of potential advanced SMR concepts remains an ongoing task. The initial concept developed was the ALMR, as documented in Reference [1]. Additional concepts are available for modeling. The molten-salt-cooled advanced high-temperature reactor (AHTR) is similar in many respects to the ALMR concept. These similarities in design construction allow for a relatively easy translation. As a result, the FHR was the next concept chosen for inclusion in the expanding library. The basic design concepts associated with the FHR are described in the next section, followed by the specific design chosen for modeling (SmaHTR). The remaining subsections include a discussion of the modification/development of the system and component models.

3.1.1 Fluoride High-Temperature Reactors

Molten-salt reactors (MSRs) are typically classified in one of two groups. There are molten salt reactor concepts in which the fuel is mixed with the primary molten salt, referred to as an MSR. Separate from these are the reactor concepts that include solid-fuel cooled by a molten salt. This second class is typically considered in the AHTR class that uses fabricated fuel. The FHR is typical of this class and is the subject of DOE design and development effort. Both the MSR and FHR concepts retain the safety and cost advantages of a low-pressure, high-temperature coolant, also shared by liquid metal reactors. In particular, the low pressure system eliminates the need for a large, expensive steel pressure vessel. Due to the higher operating temperatures, the conversion of the heat to electricity can also use an efficient, lightweight Brayton cycle gas turbine.

FHRs offer the advantage of both electrical power generation and industrial heat process generation, combined with limited environmental impact and a high degree of passive safety that decreases the likelihood of large accidents that would significantly harm the public. However, based on the current level of development, FHRs remain a longer-term power production option. A principal area of technology development that must be focused on is related to materials issues for the components and systems under design. In addition, a licensing path needs to be developed for this reactor concept that includes all the potential safety issues and design basis events. Shortening the FHR time to market includes safety evaluation and licensing approach development, as well as both concept and technology development and demonstration.

Material issues that pose major design challenges arise from the chemical nature of molten salts. Molten salts can be highly corrosive, even more so as temperatures rise. In spite of materials difficulties that higher operating temperatures present, high temperatures are desirable for increased thermodynamic efficiency, as well as potential thermochemical production of hydrogen. Early experiments at ORNL showed that Hastelloy-N and similar alloys can be used at operating temperatures up to about 700°C. The transition from an experimental reactor to a production reactor however requires longer-term experience with the behavior of the material at these temperatures, and materials at these temperatures have not yet been fully validated.

The development of system models that test the limits of normal operation will be essential in defining the bounding physical and environmental design limits that must be established.

As noted, FHRs designed specifically for electricity production are similar in system details (Fig. 6) to the liquid metal reactor concept detailed in previous reports. As a result, the SMR modeling architecture previously developed requires no fundamental changes (Fig. 7).

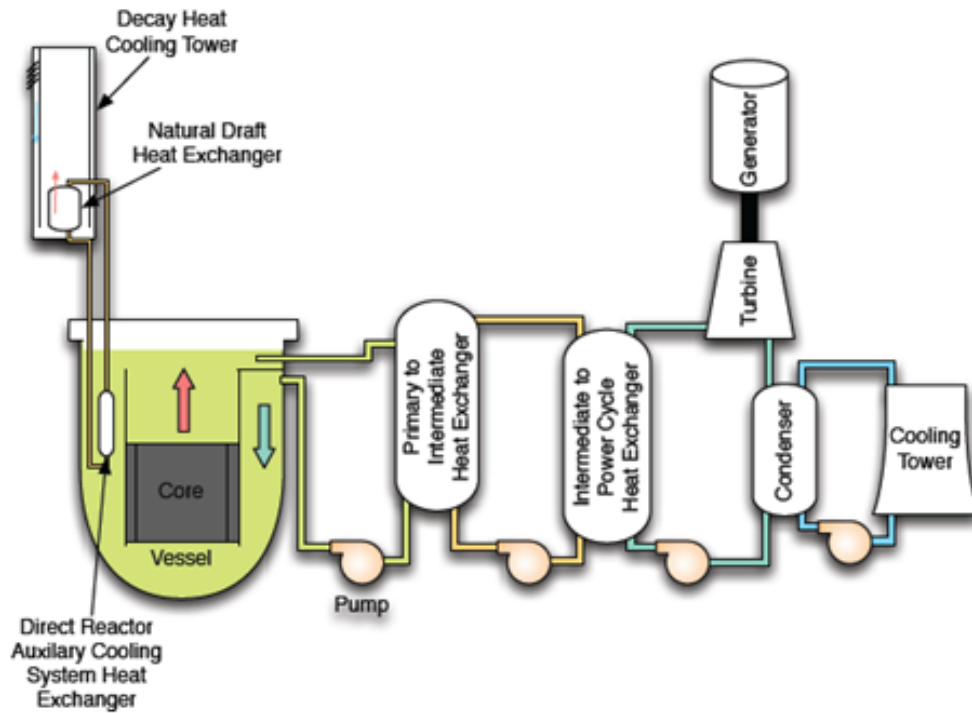


Fig. 6. Fluoride high-temperature reactor.

The end-to-end architecture for the SmaHTR is shown below. The core model is coupled to the intermediate heat exchanger (IHX) and a passive heat rejection system. The IHX connects through the IHTS and steam generator before the PCS and ultimately to the grid model for electricity production.

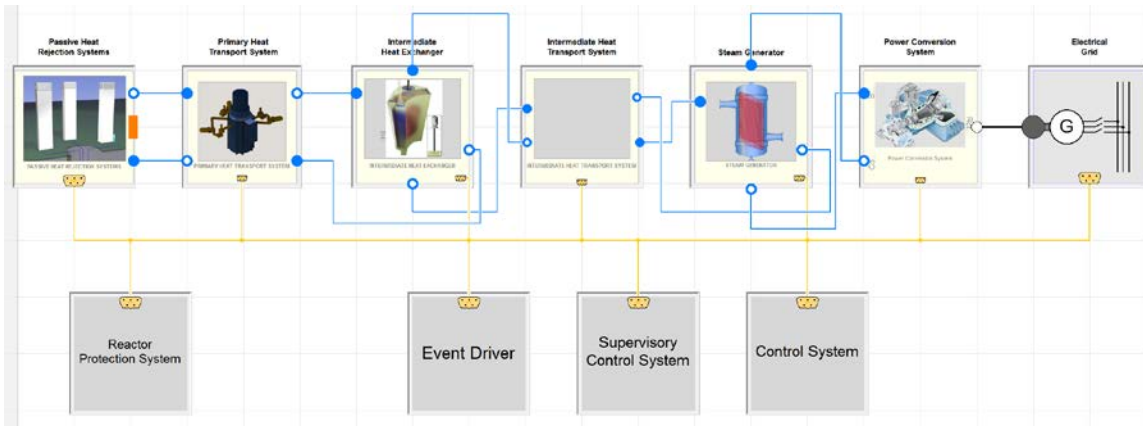


Fig. 7. Top-level architecture of the SmaHTR dynamic simulation tool for electricity generation.

Since the SmaHTR is not explicitly designed to produce electric power, an alternate end-to-end architecture (Fig. 8) is also included which can connect the steam generator to a “hydrogen production plant,” which currently serves as a placeholder for any industrial process that requires high-temperature steam. As the need arises, it can be filled by any component accepting an input of high-temperature steam and returning water to the steam generator.

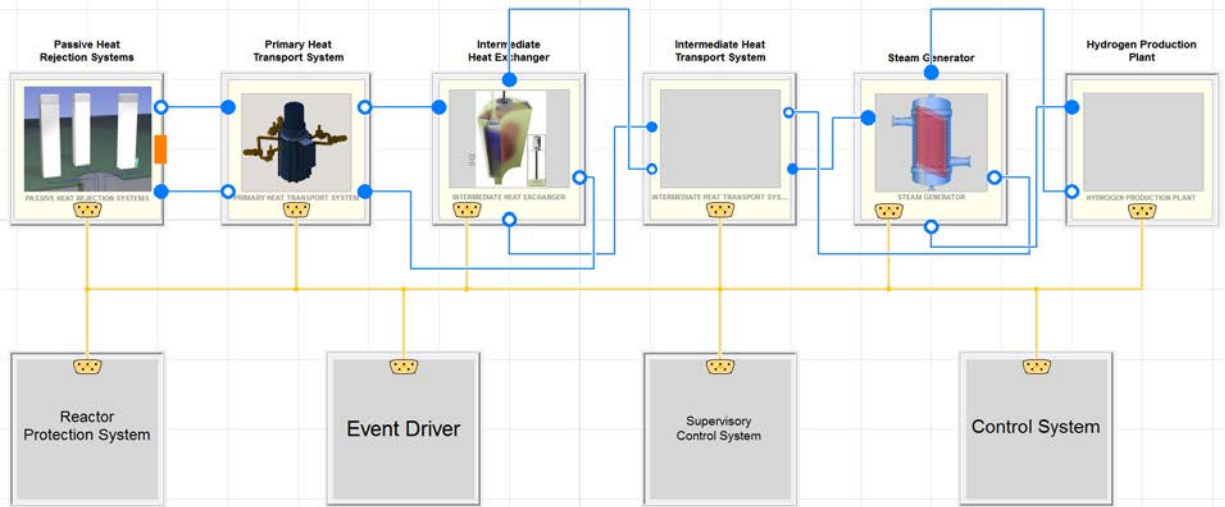


Fig. 8. Top-level architecture of the SmaHTR dynamic simulation tool for hydrogen production.

Similar to the model developed for the ALMR, the FHR is based on a conceptual design that has been proposed. This design is the SmaHTR design developed by ORNL [2]. What follows in the next section is a brief description of the concept and then a discussion of the component and systems implementations in Modelica, with a description of the similarities and differences between the ALMR and the SmaHTR concepts. Although the overall systems architecture is unchanged, the modular implementations for the various components are different because of the change in fluid and the overall operating parameters for FHR operation. In the following sections, we will discuss the various changes in these models and the different implementations that have been developed for the FHR architecture model repository.

3.1.2 SmaHTR Reactor Concept

The SmaHTR concept is based on the AHTR design but incorporates a smaller reactor module that can be used where larger power generation is not needed (or feasible) or coupled together to create an aggregate reactor system with power production scaled up as desired. The SmaHTR concept has been described in full detail in Reference [2]. The remainder of this section provides a brief description of the SmaHTR concept contained in Reference [2].

The SmaHTR concept is being designed as a system capable of providing reliable, economically attractive electricity and process heat. The potential value of such a system improves significantly as the reactor outlet temperature rises above 600°C but requires that fundamental material challenges above this temperature be addressed. In terms of electricity production, thermal-to-electric power conversion efficiencies increase from the mid 30% range at light-water-reactor (LWR) operating temperatures (~300°C) to the mid 50% range as reactor operating temperatures rise to 750°C—with still higher efficiencies as operating temperatures rise above this level.

With regard to process heat applications, numerous petrochemical refining processes require high-quality heat in the 600–700°C range. Small reactor systems operating in the 750°C range would be well suited for remote production of high-pressure steam to enable petroleum extraction from oil sands. Hydrogen production via high-temperature electrolysis and steam-methane reforming becomes practical at temperatures in the 800–850°C range (and is currently produced via natural gas combustion). The attainment of reactor core outlet temperatures of 900–1000°C would enable a variety of thermal chemical processes for the production of hydrogen from water, gasification of hard coal and lignite, etc. Thus, the development of a reliable, economical, and flexible reactor system capable of delivering heat at 600–1000°C would

revolutionize highly efficient electrical power production and the production of liquid fuels for transportation and other applications.

The SmAHTR functional requirements include:

1. Reactor power level: 125 MWt
2. Maximum operational fuel temperature: 1250°C
3. Reactor core outlet temperature: 700°C
4. Passive decay-heat removal capacity: 1% of full power
5. Reactor vessel and internals transportable via standard 53 ft commercial tractor-trailer
6. vehicles
7. Thermal-to-electric power conversion efficiency: >40%
8. System architecture and technology suite that can be adapted to higher temperatures

The SmAHTR reactor (Fig. 9) has three distinct working fluid loops: (1) the reactor primary loop, (2) the intermediate cooling loop (ICL), and (3) the DRACS decay-heat-removal loop. Additional working fluids are necessary for both the electricity generation and heat storage configuration options.

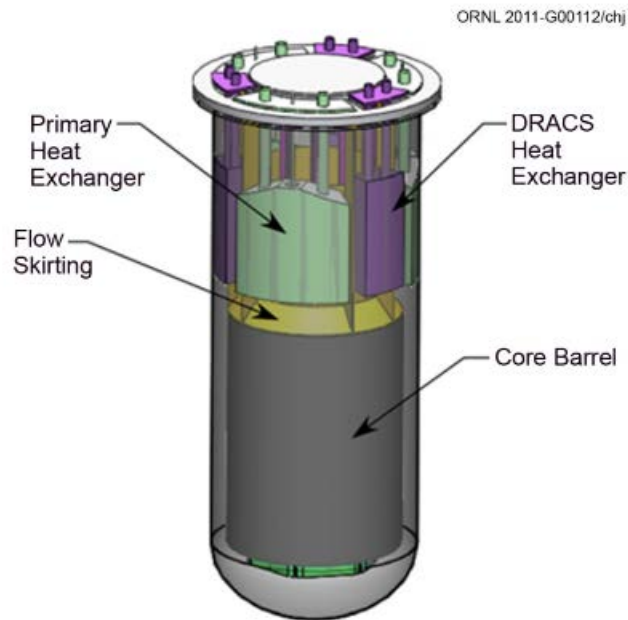


Fig. 9. SmAHTR reactor vessel (isometric).

SmAHTR employs three in-vessel primary heat exchangers (PHXs). Each PHX is coupled with a main circulating pump (MCP) that directs primary coolant salt from the common riser region above the reactor core down through the shell side of the PHX into a common downcomer region located between the outside of the core reflector and the inside of the reactor vessel. The coolant flows down through the downcomer annulus region to the lower head region of the reactor vessel and then up through the core and back to the common riser region, thus completing the main circulating loop.

Each of these in-vessel cooling loops is termed a main circulating loop (MCL). The secondary (tube) side of each PHX is an integral element of a companion intermediate cooling loop (ICL). Each ICL consists of the secondary side of the PHX, a companion ILP, and an intermediate heat exchanger (IHX) that transfers the heat to the ultimate load (either the electrical PCS or the process heat storage system and load).

During normal operations, all three MCLs and ICLs are active, each removing one-third of the heat produced by the reactor. This is accomplished by adjusting the in-vessel MCP flow and the companion ILP in each of the three ICLs.

A summary of some of the major design features is shown in Figs. 10–14. Additional details for the individual model components can be found in Sect. 3.2 or Reference [2].

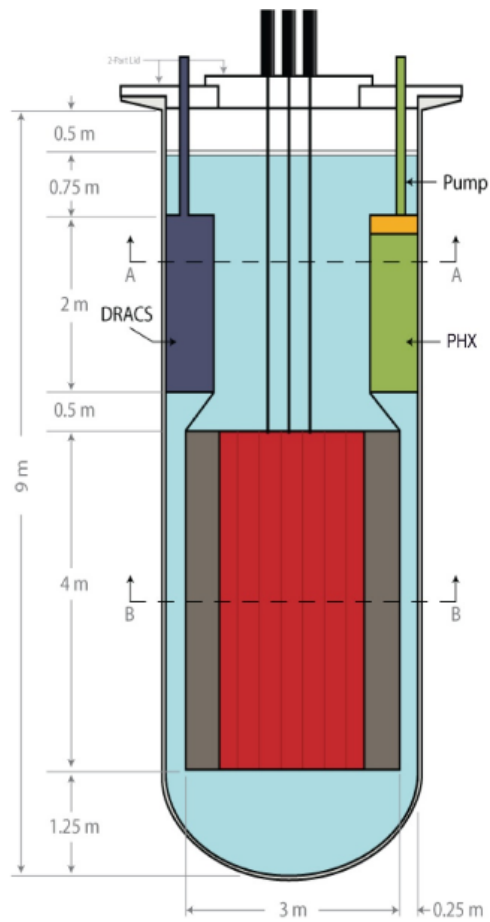


Fig. 10. SmAHTR reactor vessel (cross-section).

Variable	Value
Reactor power, MW(t)	125
Core volumetric power density, MW(t)/m ³	9.4
Primary coolant salt	FLiBe
Fuel type	TRISO, low enriched uranium
TRISO packing fraction, vol. %	50
Fuel enrichment, wt %	19.75
Core uranium loading at BOL, ^a kg	1600–2020 ^b
Core life, years	4.19
Fuel configuration	Annular pins
Fuel pin diameters (inside, outside), cm	2.2, 6.5
Fuel surface coating thickness, cm	0.3
Moderator material	Graphite
Moderator configuration	Pins and blocks
Moderator pin diameter, cm	6.16
Number of total fuel assemblies/blocks	19
Number of core assembly rings	3
Number of fuel pins/assembly	15
Number of graphite pins/assembly	4
Core height, m	4
Core effective diameter, m	~2.2
Reflector configuration and material	Radial, graphite blocks
Reflector diameter, effective inside, outside, m	~2.2, 3
Vessel height, m	9
Vessel diameter, m	3.5

Fig. 11. SmaHTR design parameters.

Variable	Value
Vessel wall thickness, cm	2.5
Vessel weight (empty, no lid), kg	22,516
Vessel and skirt material	Hastelloy-N
Secondary coolant salt	FLiNaK
Core cooling mode	Forced convection
Core flow direction	Upward
IHX/downcomer flow direction ^a	Downward
Number of main coolant pumps	3
Number of PHXs ^a	3
Number of DRACS ^a	3
PHX/DRACS annulus, height, m	2
PHX/DRACS annulus, diameter-inner, m	2.365
PHX/DRACS annulus, diameter-outer, m	3.5

^aBOL = beginning of life; IHX = intermediate heat exchanger; PHX = primary heat exchanger; DRACS = director reactor auxiliary cooling system.

^bCore uranium loading depends upon the fuel concept employed and the refueling interval requirement. The range presented encompasses all fuel concepts and refueling intervals considered to date in the trade study.

Fig. 12. SmaHTR design parameters.

Variable	Units	Value
Reactor power	MW(t)	125
Primary coolant salt		FLiBe
Flow/pump (3 operating)	kg/s	510
Flow/pump (2 operating)	kg/s	765
Vortex valve diodicity		50
Bypass flow/DRACS ^a	kg/s	68
Bypass flow, total (3 DRACS)	kg/s	205
Total core flow	kg/s	1325
Total core flow area	m ²	0.6226
Core coolant velocity	m/s	1.1
Coolant temperature at core inlet	°C	650
Core outlet coolant temperature—inner, middle, outer radial zones	°C	703, 692, 689
Coolant temperature, upper plenum (mixed)	°C	692
Coolant temperature, top plenum	°C	687
Maximum fuel temperature, centerline	°C	1027
Maximum fuel temperature, surface	°C	988
Maximum fuel surface coating temperature	°C	786
Maximum fuel heat flux	W/m ²	6.3×10^5
Fuel/Coolant heat transfer coefficient	W/(m ² · °C)	4700–6380
Core pressure drop	kPa	15
Pumping power main pump	kW	10
PHX capacity, 3 operating/2 operating ^d	MW	42/63
PHX secondary salt		FLiNaK
PHX secondary flow, each, 3 operating/2 operating	kg/s	247/370
DRACS heat losses (per DRACS)	MW	0.45
DRACS heat losses, total (3 DRACS)	MW	1.35

^aDRACS = direct reactor auxiliary cooling system; PHX = primary heat exchanger.

Fig. 13. SmaHTR design parameters.

Individual SmaHTR modules are connected together for an integrated plant, as shown in Fig. 15.

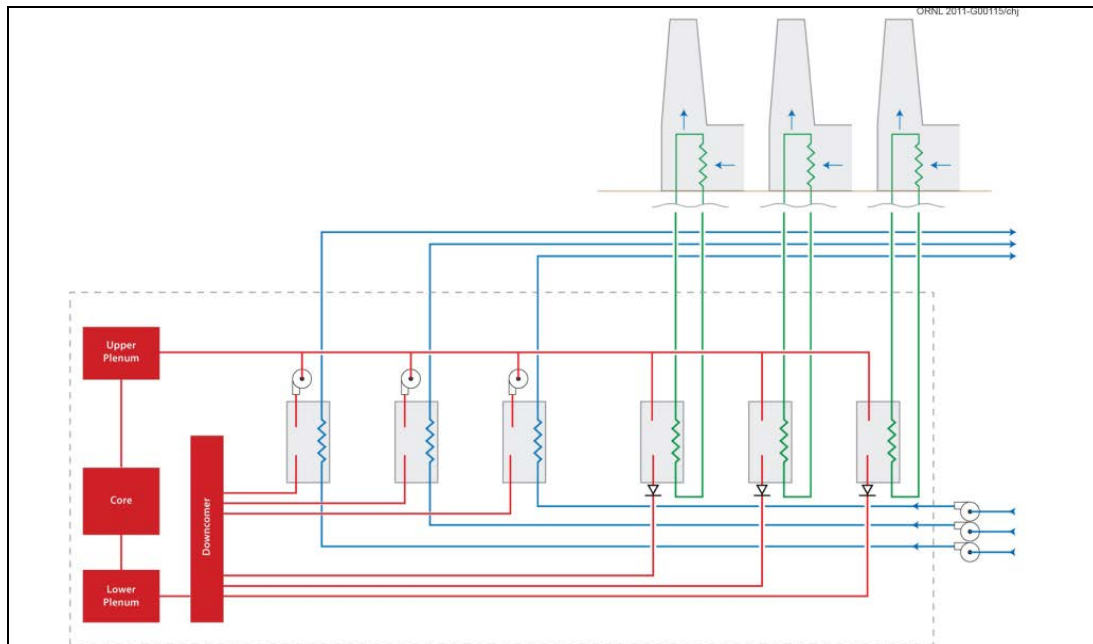


Fig. 14. SmaHTR integral systems configuration.

The development of the individual Modelica modules for this reactor concept is discussed in Sect. 3.2.

3.2 MODELICA MODEL FOR SMAHTR

3.2.1 Passive Safety System

Passive safety features for advanced reactor designs use systems or components that are based on laws of physics rather than engineered systems to ensure safe reactor shutdown and cooling without relying on active, powered systems. For the ALMR and the FHR, passive safety systems include either or both a DRACS and a RVACS. Currently the models developed include both the RVACS and the DRACS. The development of the RVACS and DRACS models as well as the Modelica implementation is covered in Sect. 2.2 of this report.

3.2.2 Reactor/Primary Heat Transport System

Reactor

The neutron dynamics is represented using the *Reactor Kinetics* class, as shown in Fig. 15. The object accepts control rod reactivity in dollars as the input variable. The coolant and fuel temperature ports are provided for coupling with the fuel thermal dynamics and coolant channel thermal-hydraulic dynamics. The *Reactor Kinetics* class generates a neutron flux axial shape distribution as the output variable. The number of nodes in the axial flux profile is provided by the user.

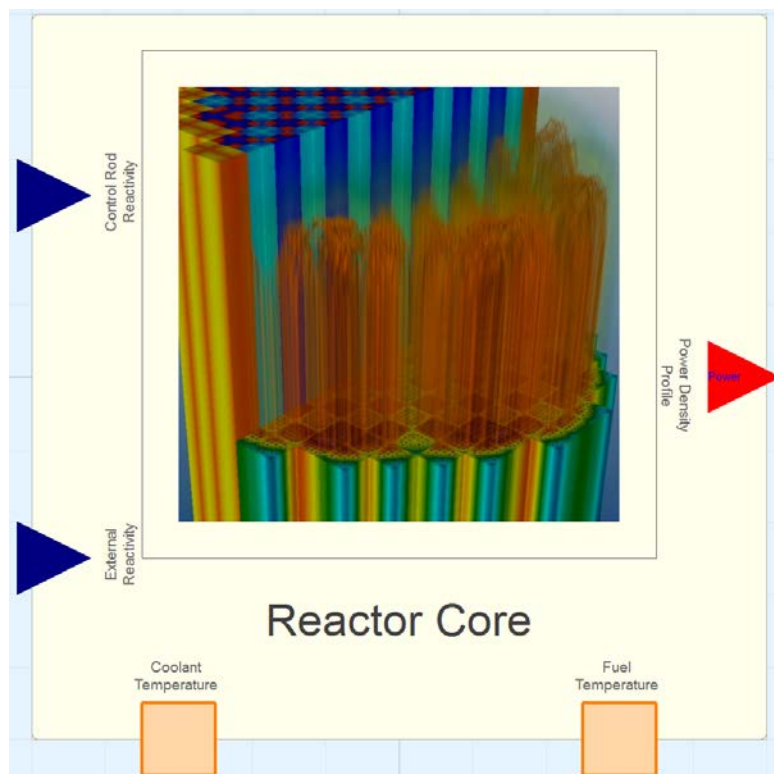


Fig. 15. Reactor kinetics model object with inputs and outputs.

The parameters of the *Reactor Kinetics* object can be provided by the user through an intuitive user interface, as shown in Fig. 16. The fuel assembly and plank geometry parameters are entered in the General tab, while reactor kinetics parameters are entered in the Kinetics tab. If no input is provided, the parameters default to neutron kinetics parameters of SmaHTR.

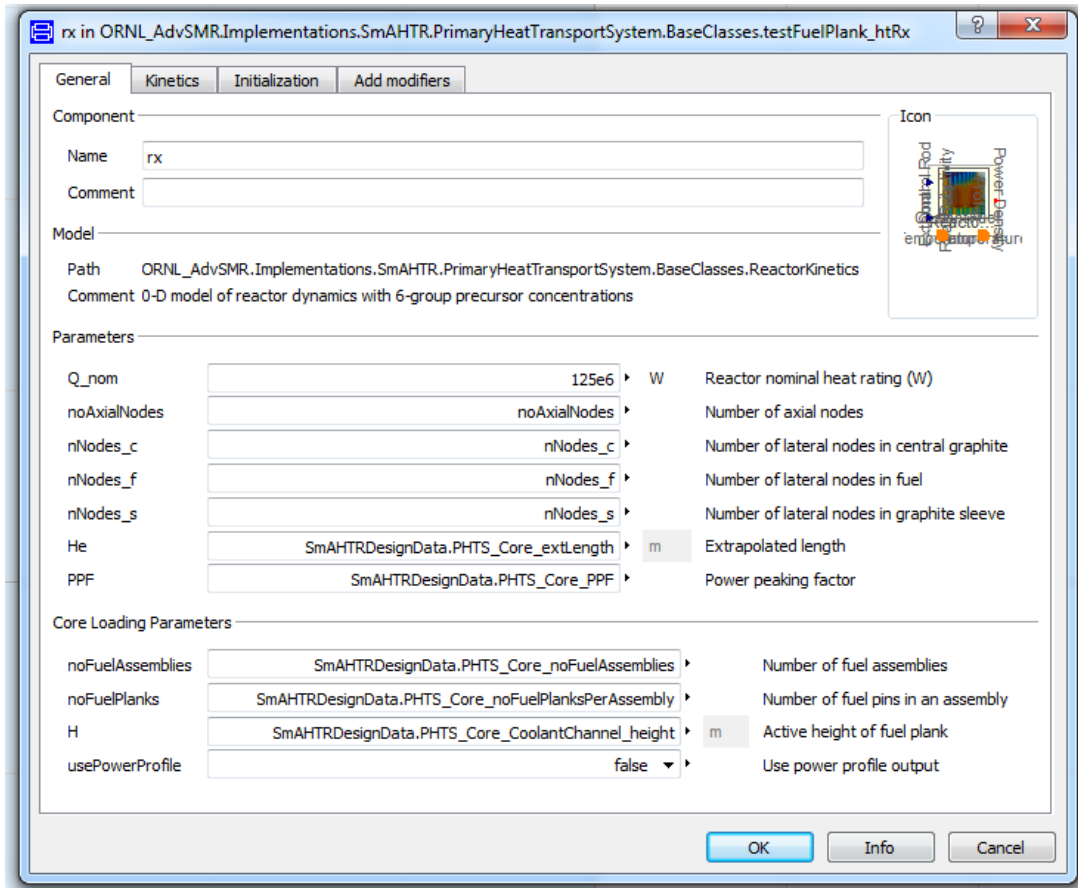


Fig. 16. User dialog for specifying geometry and neutronic parameters for the ReactorKinetics class.

Material Properties

As the FHR reactors operate at very high temperatures, temperature-dependent material properties are an important part of advanced modeling. To include this capability, the temperature-dependent properties for structural materials have been included in the Modelica libraries for the following materials.

Grade H-451 Graphite

Grade H-451 graphite is a near-isotopic, petroleum-coke-based, artificial graphite developed specifically for high-temperature gas-cooled reactor (HTGR) fuel element and reflector application. Grade H-451 graphite is used in standard fuel, reserve shutdown control (RSC) fuel, and reflector block for the next-generation nuclear plant (NGNP) design (Table 1).

Table 1. Selected thermo-physical properties of grade H-451 graphite

Property	Property Correlation ^a	
Density (kg/m ³)	1776	
	Unirradiated	$k = 169.245 - 1.24890 \times 10^{-1}T + 3.28248 \times 10^{-5}T^2$
Thermal Conductivity (W/m·K)	$0.2 \times 10^{25} \frac{n}{m^2}$	$k = 90.1445 - 3.64930 \times 10^{-2}T - 3.42932 \times 10^{-6}T^2 + 4.56817 \times 10^{-9}T^3$
	$0.5 \times 10^{25} \frac{n}{m^2}$	$k = 46.6649 - 6.75616 \times 10^{-3}T - 7.83929 \times 10^{-6}T^2 + 3.33540 \times 10^{-9}T^3$
	$1.0 \times 10^{25} \frac{n}{m^2}$	$k = 30.5337 - 1.55010 \times 10^{-3}T - 5.51300 \times 10^{-6}T^2 + 2.03348 \times 10^{-9}T^3$
	$3 - 8 \times 10^{25} \frac{n}{m^2}$	$k = 29.8193 - 7.56914 \times 10^{-3}T + 1.20901 \times 10^{-6}T^2$
Specific Heat Capacity (J/kg·K)	$c_p = -143.9883 + 3.6677 T - 0.0022T^2 + 4.6251 \times 10^{-7}T^3;$	

^aTemperature in K.

In addition to temperature-dependent material properties, the thermal conductivity of nuclear-grade graphite is also dependent on lifetime neutron fluence. At the current stage of development, there is no capability to dynamically include variation in thermal conductivity as a result of irradiation, but under the core properties, an irradiation level can be selected to investigate the effects of fuel irradiation on the larger system. The effects of irradiation on thermal conductivity are shown in Fig. 17, and the effect on core temperature is shown later. The effect of temperature on the specific heat capacity of the graphite is shown in Fig. 18.

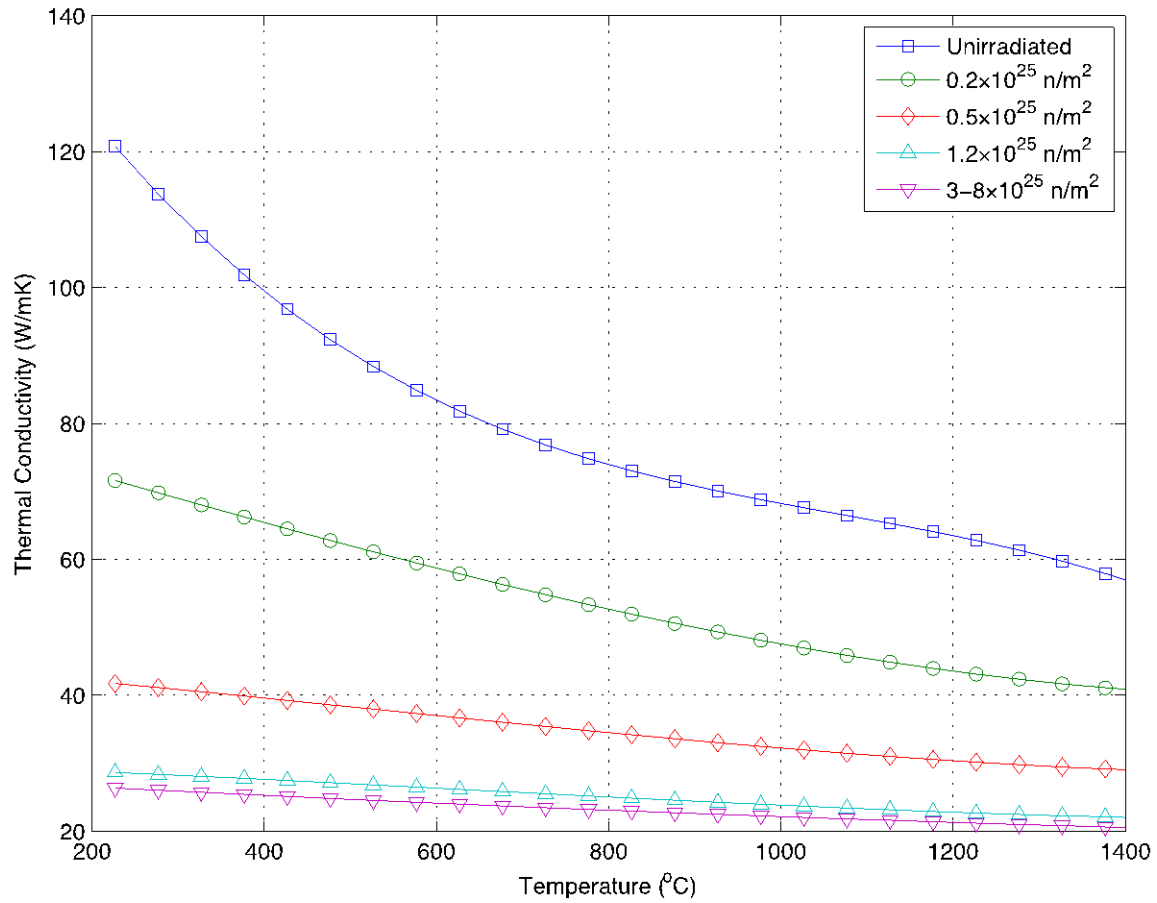


Fig. 17. Thermal conductivity of nuclear-grade graphite as a function of temperature at different fluence levels.

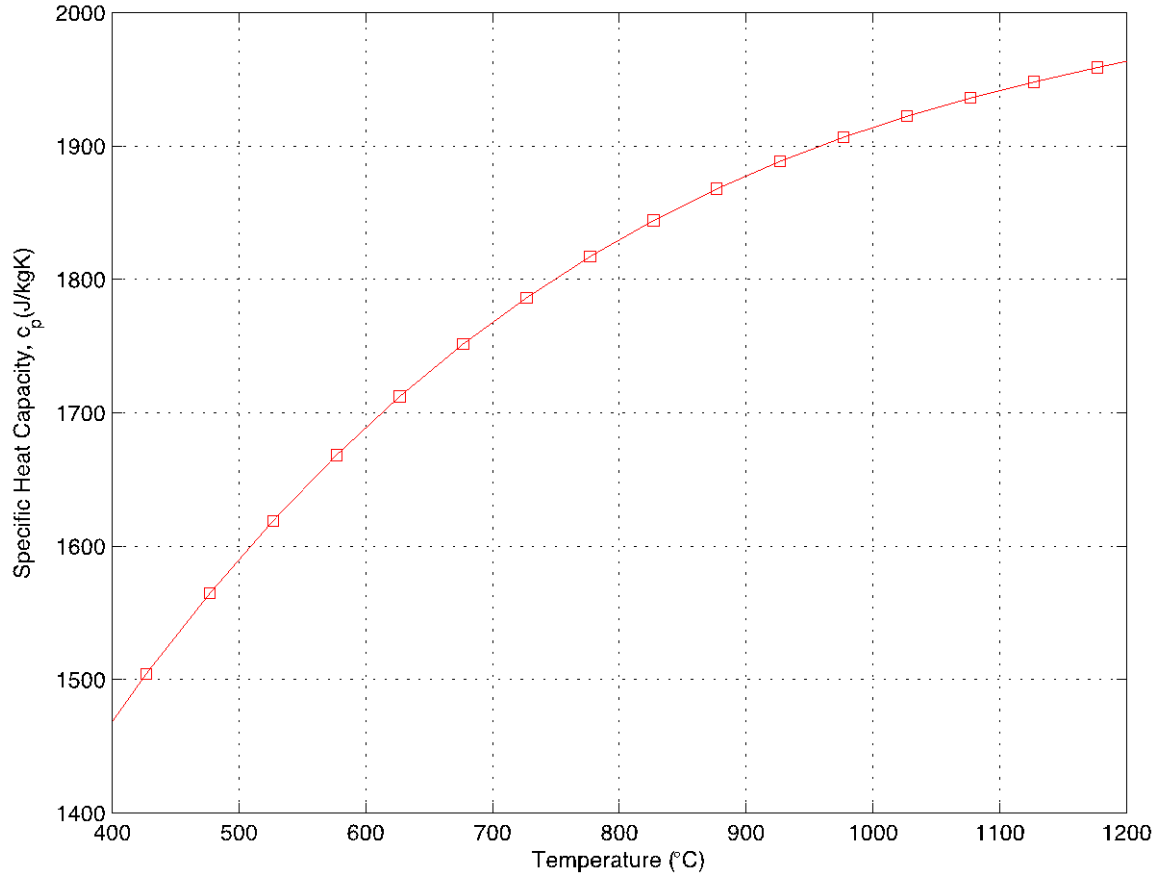


Fig. 18. Variation of specific heat capacity as a function of temperature for grade H-451 graphite.

Grade 2020 Graphite

The large rectangular grade 2020 graphite is a fine-grained, molded artificial graphite produced in large rectangular blocks. It is the reference material for permanent side reflectors and central reflector column support blocks.

TRISO in Graphite Matrix

Fuel strips contain many tri-structural isotropic (TRISO) particles dispersed in the graphite matrix. The addition of TRISO particles significantly changes the thermal properties of the matrix material, in particular its thermal conductivity. Identifying *effective* values for thermal properties is essential for proper calculation of temperature distribution in the fuel plank. A great deal of research has been done to correlate the effective thermal conductivity of the fuel region. Based upon this research, correlations have been developed in which effective thermal conductivity, k_e , of the fuel strip can be calculated using a correction factor, that is,

$$k_e = f_p k ,$$

where f_p is the packing fraction factor (also called particle volume fraction factor) defined as

$$f_p = \frac{1 - \phi}{1 + \frac{\phi}{2}} ,$$

where φ is the *packing fraction* of TRISO particles within the fuel strip.

Jensen et al. Reference [3] proposed a more accurate formulation of effective thermal conductivity:

$$k_e = \frac{3k_m k_p \varphi + (2k_m + k_p)k_m(1 - \varphi)}{3k_m \varphi + (2k_m + k_p)(1 - \varphi)},$$

where k_m is thermal conductivity of matrix and k_p is thermal conductivity of TRISO particles. This is the recommended correlation for NGNP TRISO fuels.

Hastelloy-N

Hastelloy-N is a nickel-based alloy that was invented at ORNL as a container material for liquid fluoride salts (Table 2).

Table 2. Thermal properties of Hastelloy-N

Physical property	Temperature (°C)	Quantity	Unit
Density	22	8860	kg/cm ³
	300	14.4	
	400	16.5	
Thermal Conductivity	500	18.0	W/m·K
	600	20.3	
	700	23.6	
	300	456	
	400	469	
	480	477	
	540	485	
Specific Heat Capacity	570	523	J/kg·K
	590	565	
	620	586	
	660	582	
	680	578	
	700	578	
	Coefficient of Thermal Expansion	300	
400		12.7 x 10 ⁻⁶	
500		13.4 x 10 ⁻⁶	
600		14.0 x 10 ⁻⁶	
700		15.3 x 10 ⁻⁶	

Experimental data for thermal conductivity of Hastelloy-N are plotted in Fig. 19. Also shown in the plot is the linear fit to the data between 400 and 700°C, which yields the following expression:

$$k(T) = 6.62 + 0.0236 T .$$

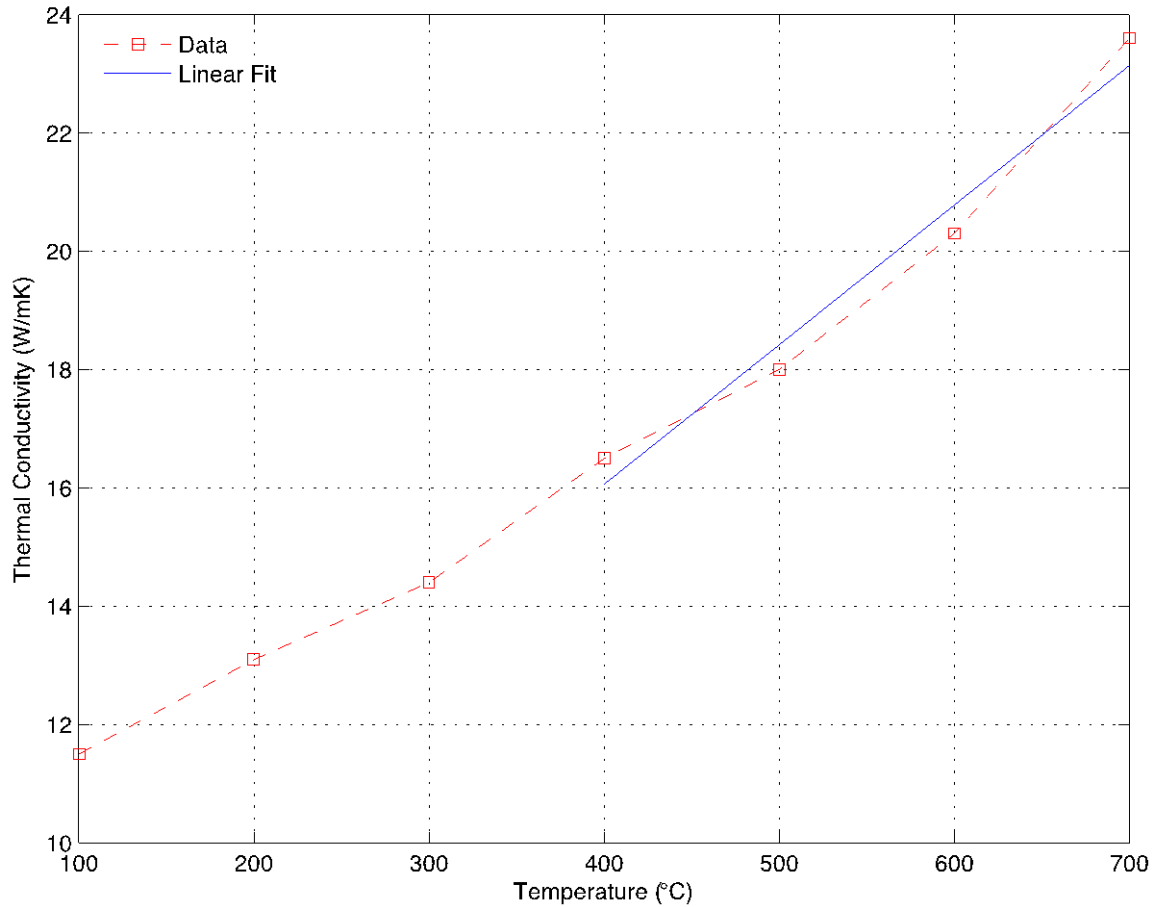


Fig. 19. Thermal conductivity of Hastelloy-N as a function of temperature [HAYNES –Reference[4]].

Specific heat capacity exhibits more dramatic changes in the temperature range of interest. Experimental data is plotted in Fig. 20 along with the linear fit made for the temperature range between 400 and 700°C. The linear fit for specific heat capacity is given by the expression below.

$$c_p(T) = 269.66 + 0.4611 T .$$

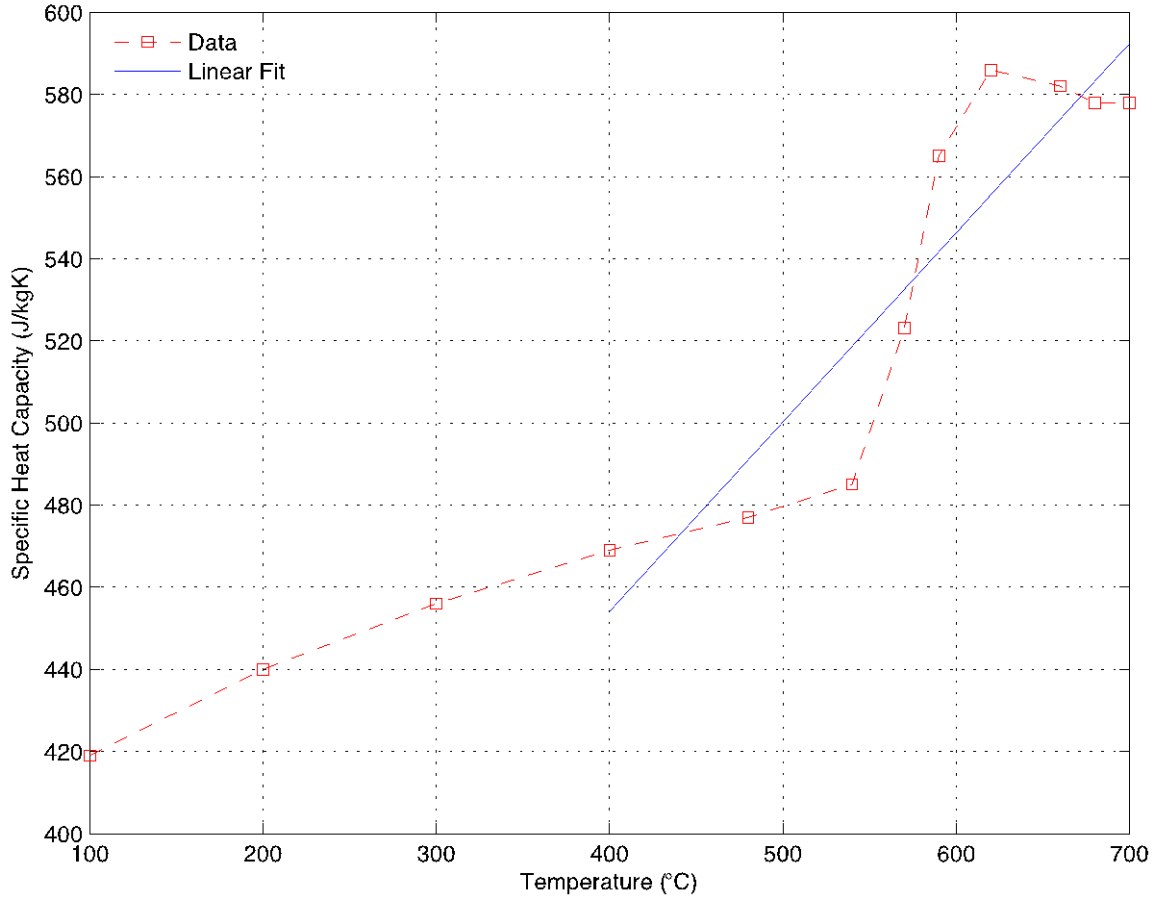


Fig. 20. Specific heat capacity of Hastelloy-N as a function of temperature [HAYNES-Reference[4]].

Reactor Kinetics

The prompt portion of normalized heat generation is implemented with the *point kinetics* equations.

$$\frac{dn}{dt} = \frac{\rho_t - \beta}{\Lambda} n(t) + \frac{1}{\Lambda} \sum_{i=1}^6 \beta_i c_i(t) \quad \text{and} \quad (1.a)$$

$$\frac{dc_i}{dt} = \lambda_i [n(t) - c_i(t)] \quad , \quad (1.b)$$

where $n(t)$ is normalized reactor power, $c_i(t)$ is the normalized concentration of the i -th group delayed neutron precursor, β_i is the fraction of the i -th group precursor, λ_i is the decay constant for the i -th group precursor, $\rho_t(t)$ is the total reactivity, and Λ is mean neutron generation time. The rate equations are subject to steady-state initial conditions, that is,

$$\dot{n} = \dot{c}_i = 0 \quad .$$

The delayed portion of normalized heat generation is implemented using Eq. (2)

$$Q_{n-decay} = 0.1 \left[(t+10)^{-\frac{1}{5}} - (t+T_s)^{-\frac{1}{5}} + 0.87(t+T_s+2 \times 10^7)^{-\frac{1}{5}} - 0.87(t+2 \times 10^7)^{-\frac{1}{5}} \right], \quad (2)$$

where t is time after shutdown and T_s is the operation time prior to shutdown—both in seconds.

The reactivity feedbacks are modeled as follows:

$$\rho_f = \alpha_f (T_{fe} - T_{f0}), \quad (3.a)$$

$$\rho_c = \alpha_c (T_{ce} - T_{c0}), \text{ and} \quad (3.b)$$

$$\rho_t = \rho_{CR} + \rho_f + \rho_c, \quad (3.c)$$

where ρ_f is the fuel Doppler reactivity feedback, ρ_c is the coolant density reactivity feedback, and ρ_t is the total reactivity feedback.

The axial neutron flux is considered to have a cosine shape defined as

$$\varphi(z) = \varphi_{max} \cos\left(\pi \frac{z}{H}\right), \quad (4)$$

where H is the active core length.

In order to account for axial leakage, Eq. (4) should be modified to account for extrapolated length, which leads to a *chopped-cosine distribution*, that is,

$$\varphi(z) = \varphi_{max} \cos\left(\pi \frac{z - \frac{H}{2}}{H_\varepsilon}\right), \quad (5)$$

where $H_\varepsilon = H + 2\varepsilon$ is the extrapolated height of the core, and ε is the extrapolation distance at the top and the bottom of the active core region.

Similarly, the power density profile, $q'''(z)$, is proportional to neutron flux profile, that is,

$$q'''(z) = q'''_{max} \cos\left(\pi \frac{z - \frac{H}{2}}{H_\varepsilon}\right). \quad (6)$$

Obviously, Eqs. (5) and (6) are acceptable forms for analytical calculations. For nodal computations where local values of variables are averaged over a finite domain, it should be discretized:

$$\langle q_i''' \rangle = \frac{\int_{z_{i-1}}^{z_i} q_{max}''' \cos\left(\pi \frac{z-H}{H_e}\right) dz}{\Delta z_i}, \quad (7)$$

where $\Delta z_i = z_i - z_{i-1}$ is the axial node i . For a uniform mesh size, as adopted in this derivation, the value becomes $\Delta z_i = H/N$ for a total number of N nodes.

Taking the integral in Eq. (7) leads to the following expression for node-averaged power density:

$$\langle q_i''' \rangle = q_{max}''' \frac{NH_e}{\pi H} \left\{ \sin\left[\pi \frac{H}{H_e} \left(\frac{i}{N} - \frac{1}{2}\right)\right] - \sin\left[\pi \frac{H}{H_e} \left(\frac{i-1}{N} - \frac{1}{2}\right)\right] \right\}, \quad (8)$$

where q_{max}''' is the ratio of maximum axial power to average power defined as

$$q_{max}''' = f_{pp} \langle q''' \rangle, \quad (9)$$

where f_{pp} is the *power peaking factor*, and $\langle q''' \rangle$ is the core average power density. In previous studies, f_{pp} was taken between 1.3 and 1.4 [REF [5-6]. The value of $f_{pp} = 1.3$ is used as the default value, but the value can be changed through the user interface.

A continuous and discretized power density profile is plotted in Fig. 21 as a function of axial position.

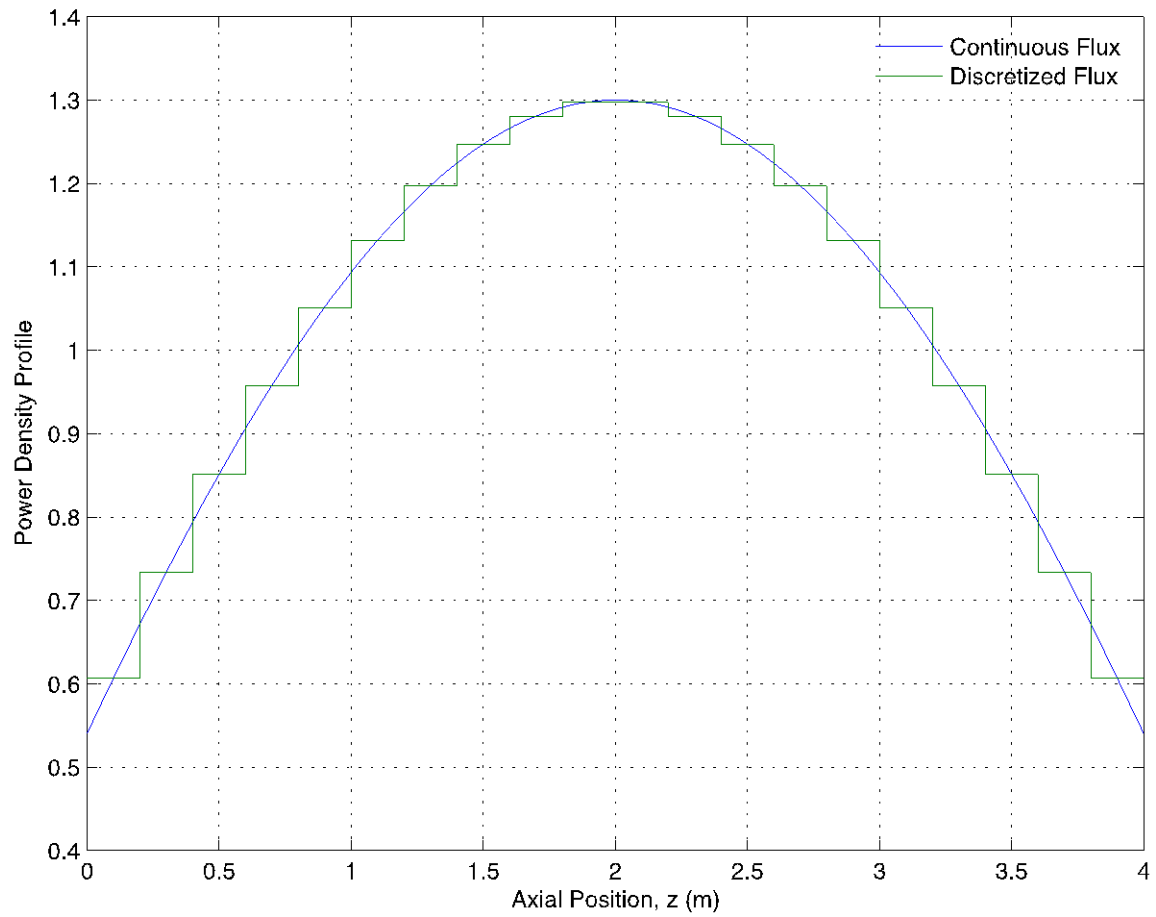


Fig. 21. Power density profile as a function of axial position.

Fuel Thermal Dynamics

The fuel equations are described below, and the geometry can be seen in Figs. 22–24.

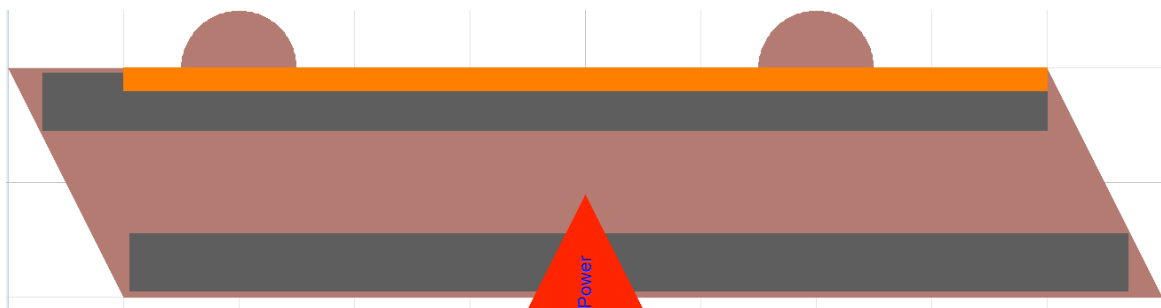


Fig. 22. Thermal conduction model block for SmAHTR fuel.

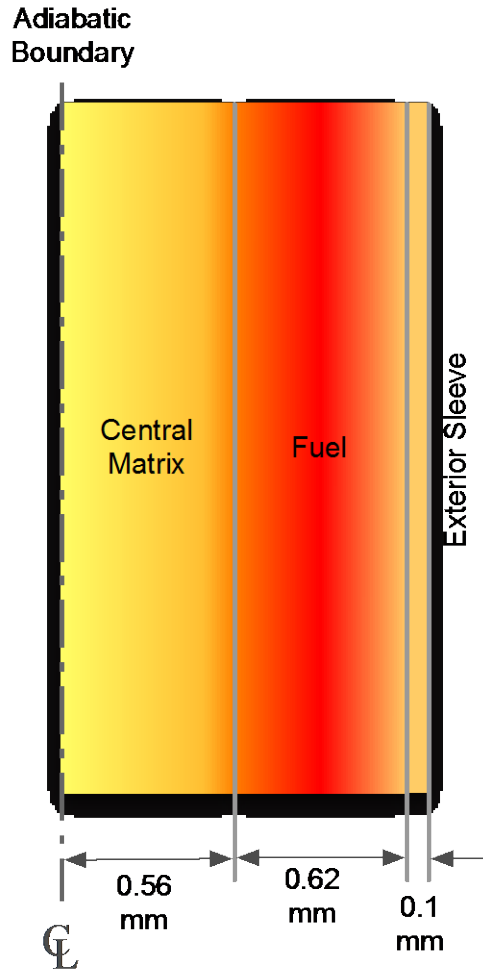


Fig. 23. Dimensions for the fuel thermal conduction model (adiabatic boundary represents the symmetry condition).

Central graphite and
Graphite sleeve regions

$$\rho c_p \frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2} \quad (9.a)$$

Fuel region

$$\rho c_p \frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2} + \dot{q} \quad (9.b)$$

subject to the following boundary conditions:

Neumann
(adiabatic boundary)

$$\frac{\partial T}{\partial x} = 0 \text{ at } x = 0 \quad (10.a)$$

Robin
(convective boundary)

$$-k \frac{\partial T}{\partial x} = h(T - T_{\infty}) \text{ at } x = L \quad (10.b)$$

$$T_1 = T_2 \quad (10.c)$$

Interface boundary

$$k_1 \frac{\partial T_1}{\partial x} = k_2 \frac{\partial T_2}{\partial x} \quad (10.d)$$

where T_1 and T_2 refer to adjacent regions.

These coupled equations form a complete set; however, the Dymola solver does not handle partial differential equations. Hence, the spatial dependence is discretized using a finite-difference formulation as follows.

Central graphite and
Graphite sleeve regions

$$\rho_i c_{p,i} \frac{dT_i}{dt} = k_i \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2} \quad (11.a)$$

Fuel region

$$\rho_i c_{p,i} \frac{dT_i}{dt} = k_i \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2} + \dot{q} \quad (11.b)$$

As can be seen in Eqs. (11.a) and (11.b), the time differential term $\frac{dT}{dt}$ is left as is, as time derivatives are handled internally by Dymola; therefore, no discretization is needed in time. It should be noted that if an adaptive solver scheme is used, such as that supported by the DASSL* or CVODE† suites, integration time steps are automatically adjusted and refined by the solver where the time rate of change of temperature becomes too steep.

* DASSL is designed for the numerical solution of implicit systems of differential/algebraic equations (DAE) written in the form $F(t,y,y')=0$, where F , y , and y' are vectors, and initial values for y and y' are given. Systems of DAE arise in several diverse applications in the physical world. Problems of this type occur frequently in the numerical method-of-lines treatment of partial differential equations, in the simulation of electronic circuits, and in the dynamic analysis of mechanical systems. The derivatives y' are approximated by backward differentiation formulae (BDF), and the resulting nonlinear system at each time-step is solved by Newton's method.

DASSL integrator was developed by Sandia National Laboratories.

† CVODE is a solver for stiff and nonstiff ordinary differential equation (ODE) systems (initial value problem) given in explicit form $y' = f(t,y)$. The methods used in CVODE are variable-order, variable-step multistep methods. For nonstiff problems, CVODE includes the Adams-Moulton formulas, with the order varying between 1 and 12. For stiff problems, CVODE includes the BDFs in so-called fixed-leading coefficient form, with order varying between 1 and 5. For either choice of formula, the resulting nonlinear system is solved (approximately) at each integration step. For this, CVODE offers the choice of either functional iteration, suitable only for nonstiff systems, and various versions of Newton iteration.

CVODE is part of the SUNDIALS suite developed by Lawrence Livermore National Laboratory.

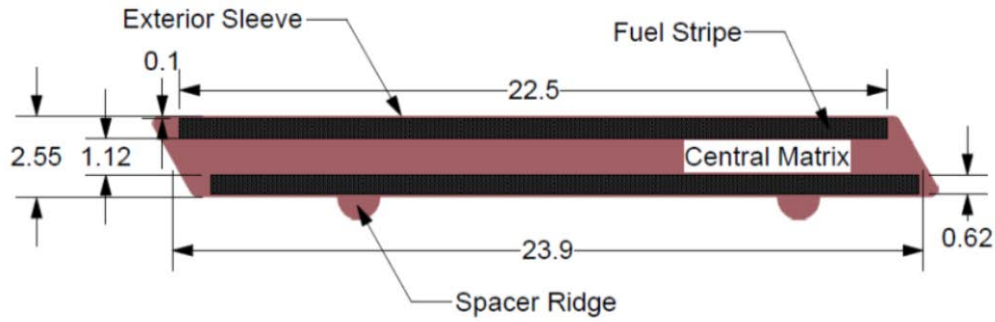


Fig. 24. Geometry of a fuel pin for thermal model (quantities in mm).

As a demonstration of the fuel model functionality and the effect of fuel irradiation on temperature, a SmAHTR core was simulated with BOL, or beginning of life (unirradiated) and EOL, or end of life ($8 \times 10^{25} \text{ n/m}^2$) fuel. The following plots (Figs. 25–28) are from the output of the core model described above.

For BOL fuel, the centerline fuel temperature is approximately 780°C (Fig. 28) and the small temperature gradient between clad and centerline temperature can be seen. It is reassuring to observe that the general temperature profiles match what would be expected in general for a cosine power distribution with coolant flow.

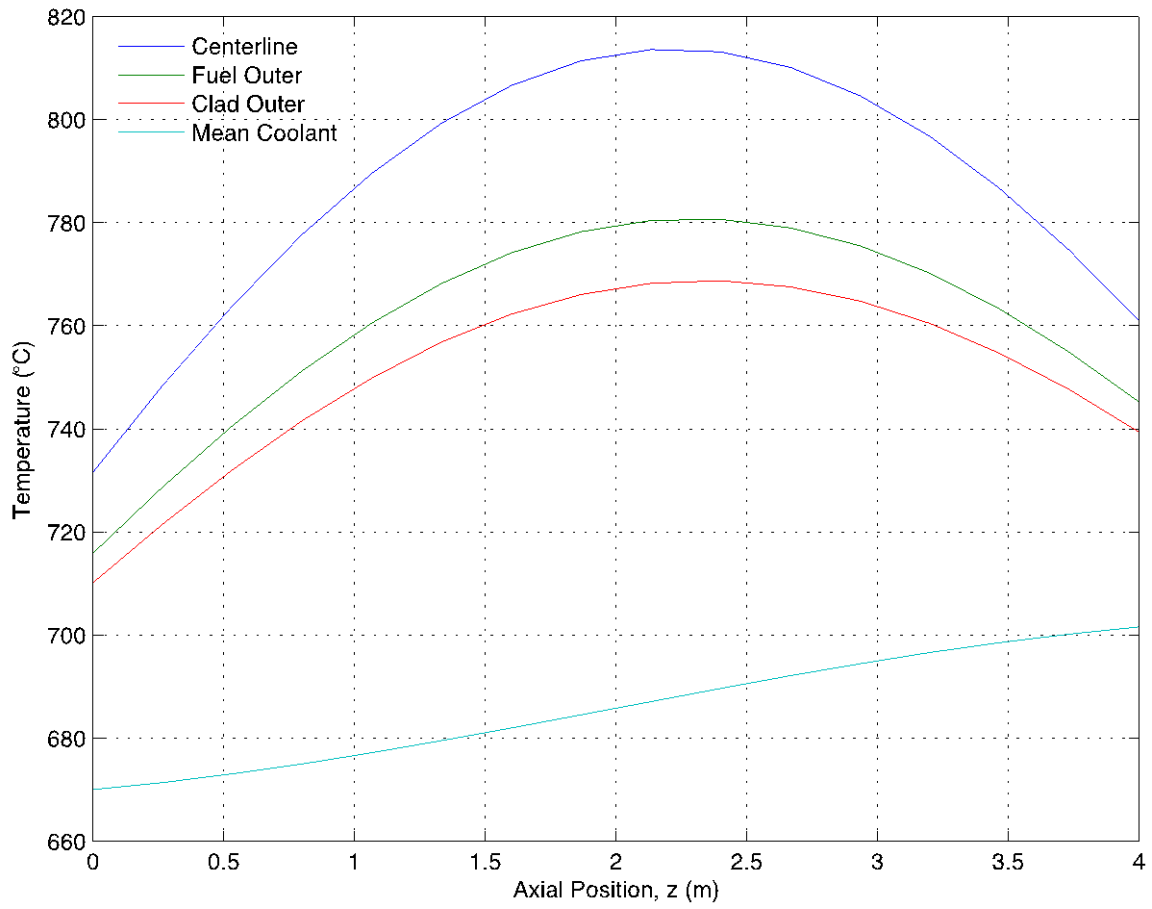


Fig. 25. Fuel and coolant axial temperature profiles for fully irradiated fuel.

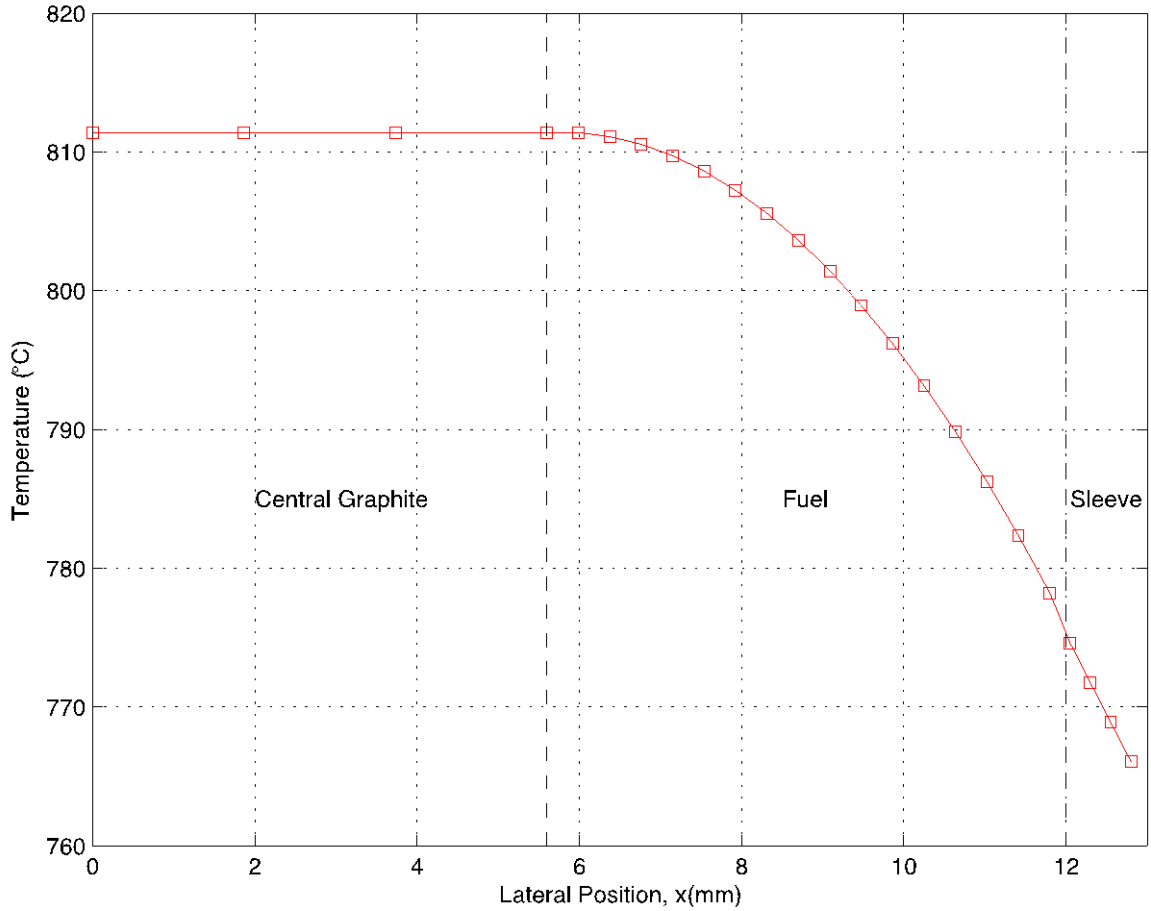


Fig. 26. Lateral fuel temperature profile at midplane for fully irradiated fuel.

For EOL fuel at the same power level (Fig. 25), the centerline temperature is somewhat greater, as well as the temperature gradient across the fuel, since the thermal conductivity is decreased by high neutron fluence. However, for EOL and BOL values, the model predictions are still in the range of what has been predicted for SmaHTR.

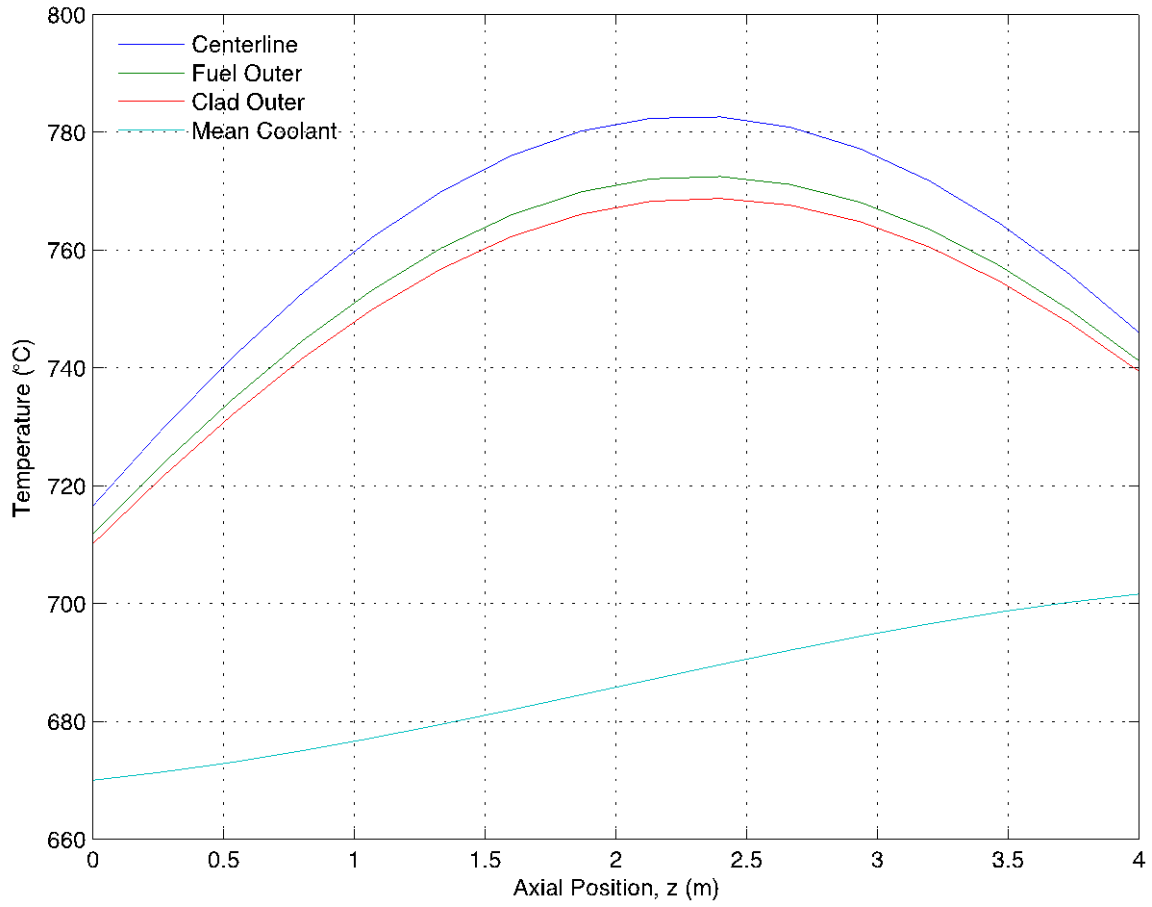


Fig. 27. Fuel and coolant axial temperature profiles for unirradiated fuel.

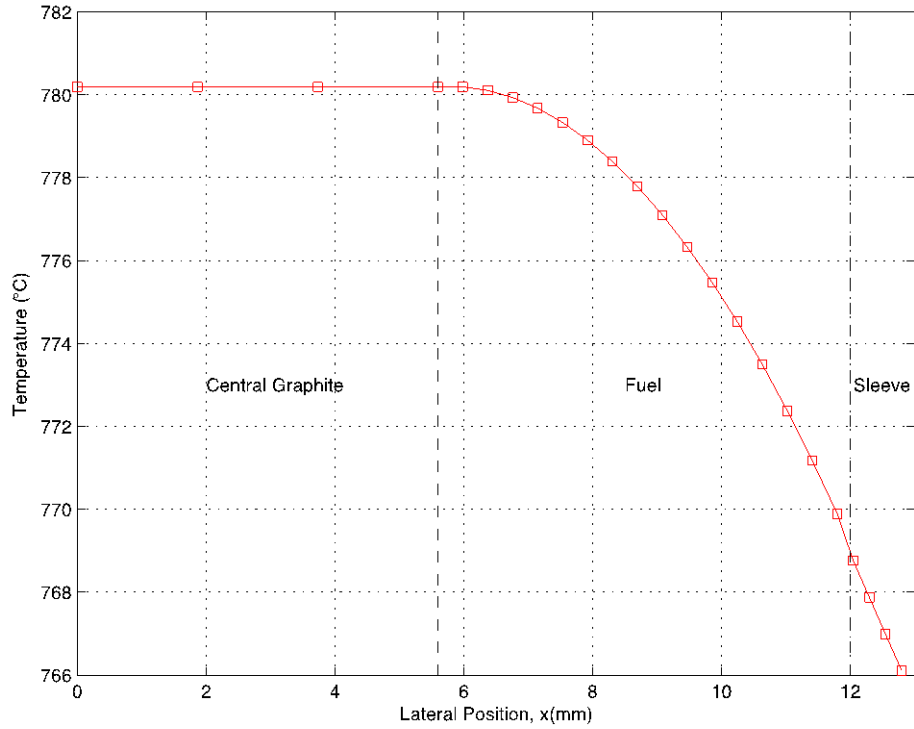


Fig. 28. Lateral fuel temperature profile at midplane for unirradiated fuel.

Coolant Channel

The fuel geometry for AHTR and SmaHTR is composed of hexagonal elements with individual fuel plates arranged within a graphite structure (Fig. 29).

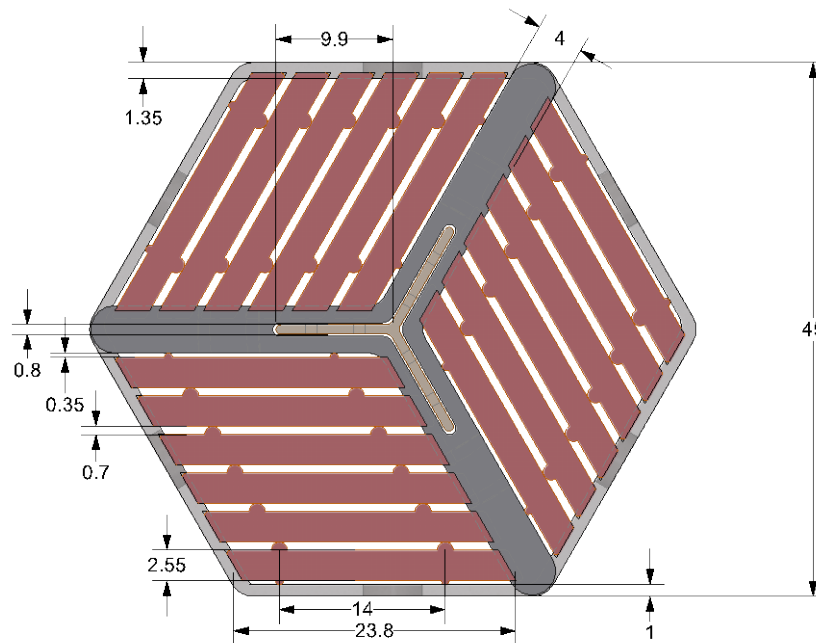


Fig. 29. Geometry of a SmaHTR assembly.

The primary coolant flows in the gaps between the trapezoidal fuel elements. The fuel channel used in the model is shown in Fig. 30.

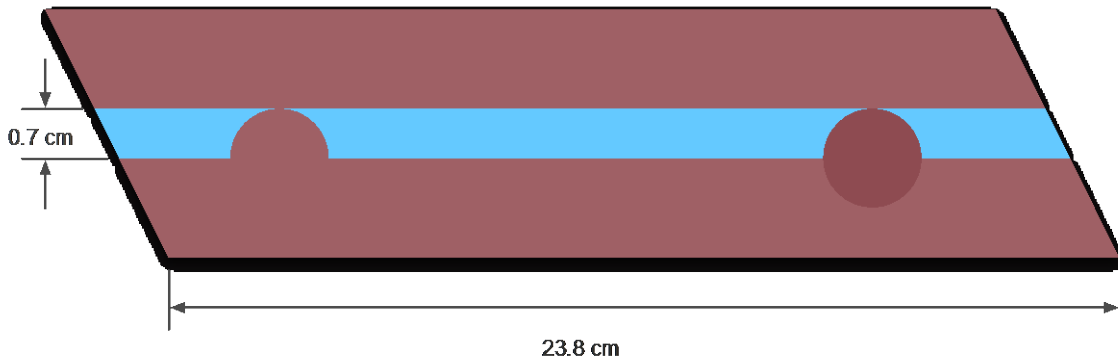


Fig. 30. A single flow channel.

The temperature distribution in the coolant and the nodal heat transfer coefficient at the nominal power level in the core are shown in the Figs. 31 and 32, respectively.

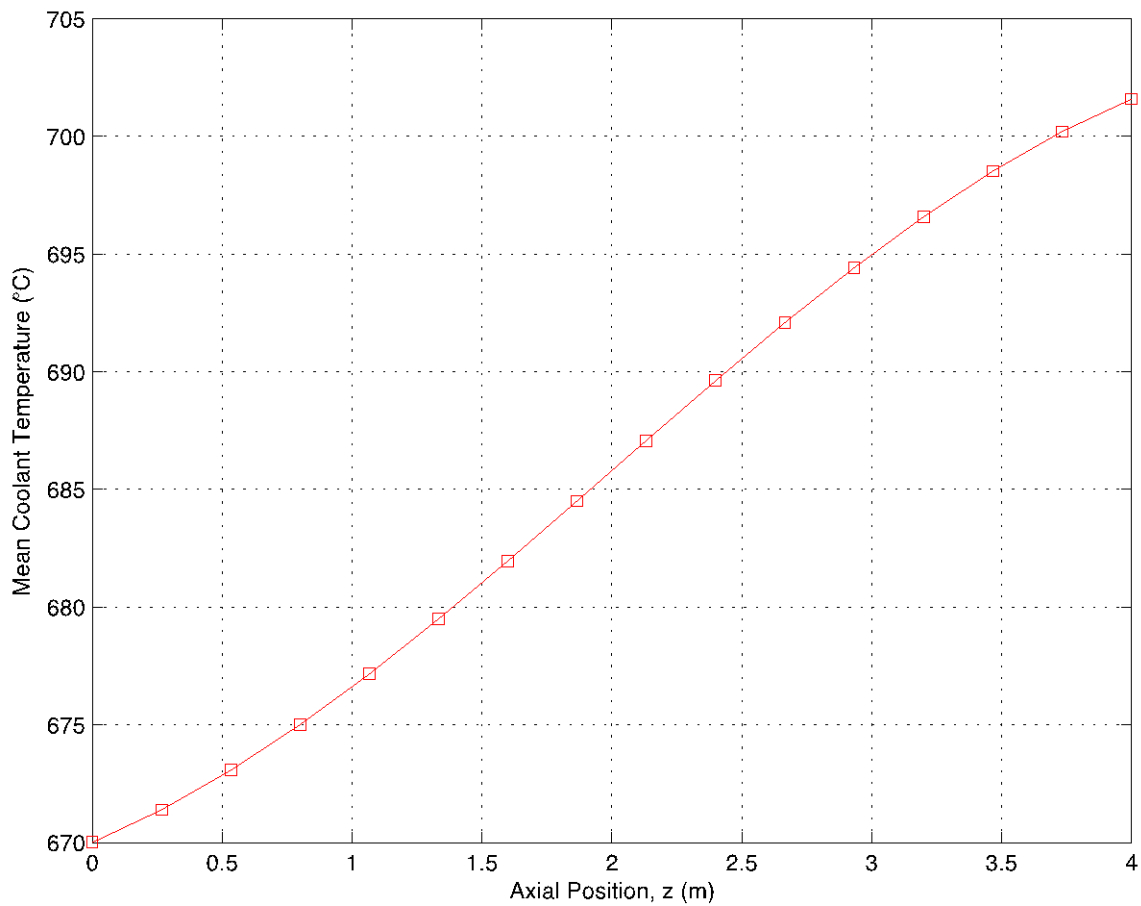


Fig. 31. Nodal mean coolant temperature profile along the channel.

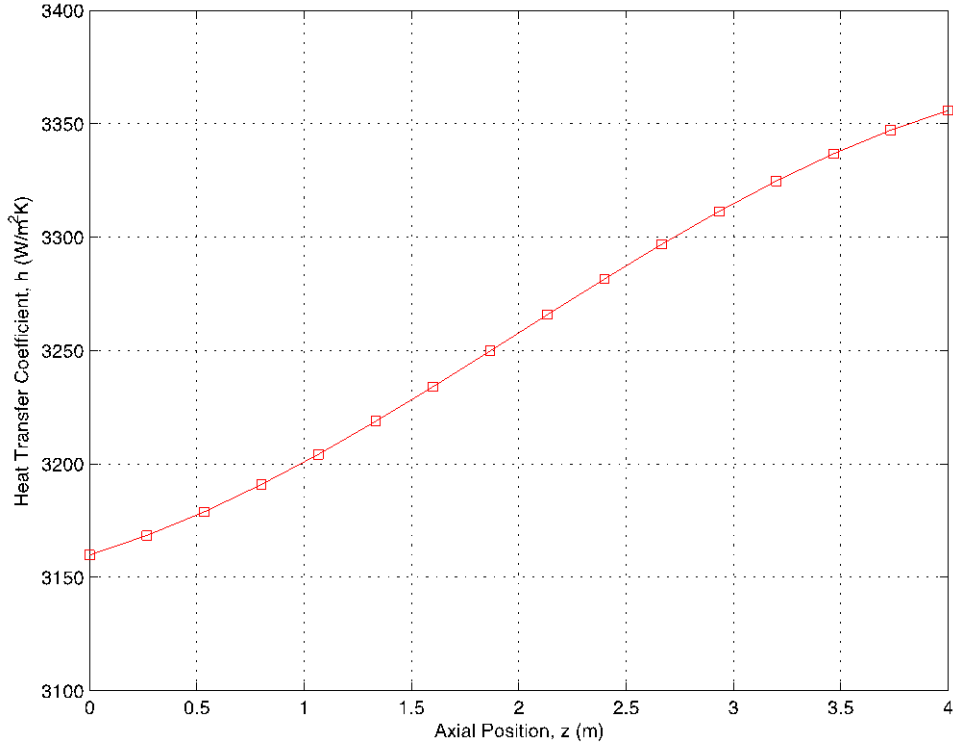


Fig. 32. Nodal heat transfer coefficient profile along the channel.

3.3 PRIMARY HEAT TRANSPORT SYSTEM

The reactor core is coupled with a primary coolant system. This coolant system is similar to the system developed for the ALMR described in Reference [1]. The principal difference is associated with the fluid and material properties and the incorporation of multiple zone regions for the core (Figs. 33–34).

Three liquid salts are considered in the FHR architecture: flibe, flinak, and KF-ZrF₄. Property libraries have been implemented for the FHR simulation package. Tables 3–5 provide the parameter used to define the fluid libraries.

Table 3. Flibe thermal hydraulic parameters

Physical property	Correlation	Unit
Density	$\rho_T = 2412.3 - 0.4884 T$	ρ_T (kg/m ³) T (K)
Specific heat capacity	$c_p = 2386$	c_p (J/kg·K)
Thermal conductivity	$k = 1.1$	k (W/m·K)
Dynamic viscosity	$\mu = 1.16 \times 10^{-4} e^{\frac{3755}{T}}$	T (K)

Table 4. Flinak thermal-hydraulic parameters

Physical property	Correlation	Unit
Density	$\rho_T = 2579.3 - 0.624 T$	ρ_T (kg/m ³) T (K)
Specific heat capacity	$c_p = 1884.1$	c_p (J/kg·K)
Thermal conductivity	$k = 0.92$	k (W/m·K)
Dynamic viscosity	$\mu = 0.4 \times 10^{-4} e^{\frac{4170}{T}}$	T (K)

Table 5. KF-ZrF₄ thermal-hydraulic parameters

Physical property	Correlation	Unit
Density	$\rho_T = 3416.0 - 0.887 T$	ρ_T (kg/m ³) T (K)
Specific heat capacity	$c_p = 1051.0$	c_p (J/kg·K)
Thermal conductivity	$k = 0.45$	k (W/m·K)
Dynamic viscosity	$\mu = 0.159 \times 10^{-4} e^{\frac{3179}{T}}$	T (K)

Initially a single zone implementation of the SmAHTR core was developed, which homogenizes the radial power distribution and calculates heat transfer over a single channel, from a single fuel element (Fig. 33).

An extension of the single channel architecture is a two-zone primary system (Fig. 34), where the flow is split between a central and radial region and the core power is distributed between those two zones. Theoretically, this can eventually be extended to more radial distribution zones or even individual fuel elements if the need arises.

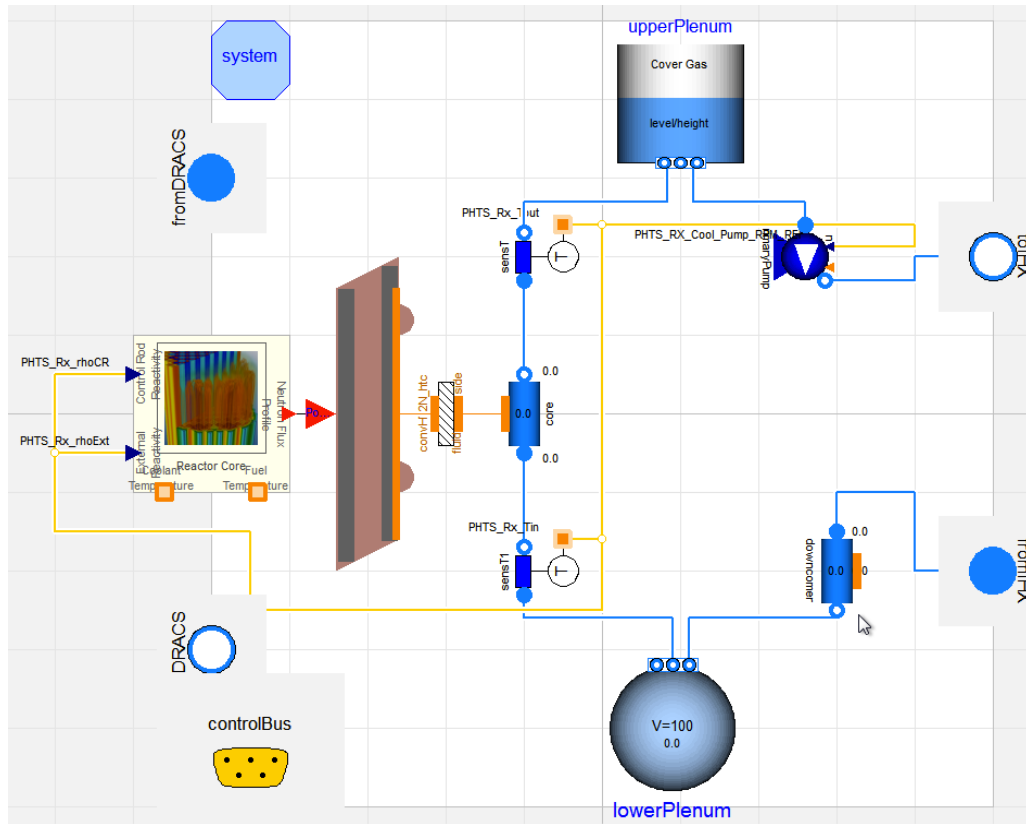


Fig. 33. Single-zone representation of the reactor core and the connected fluid systems.

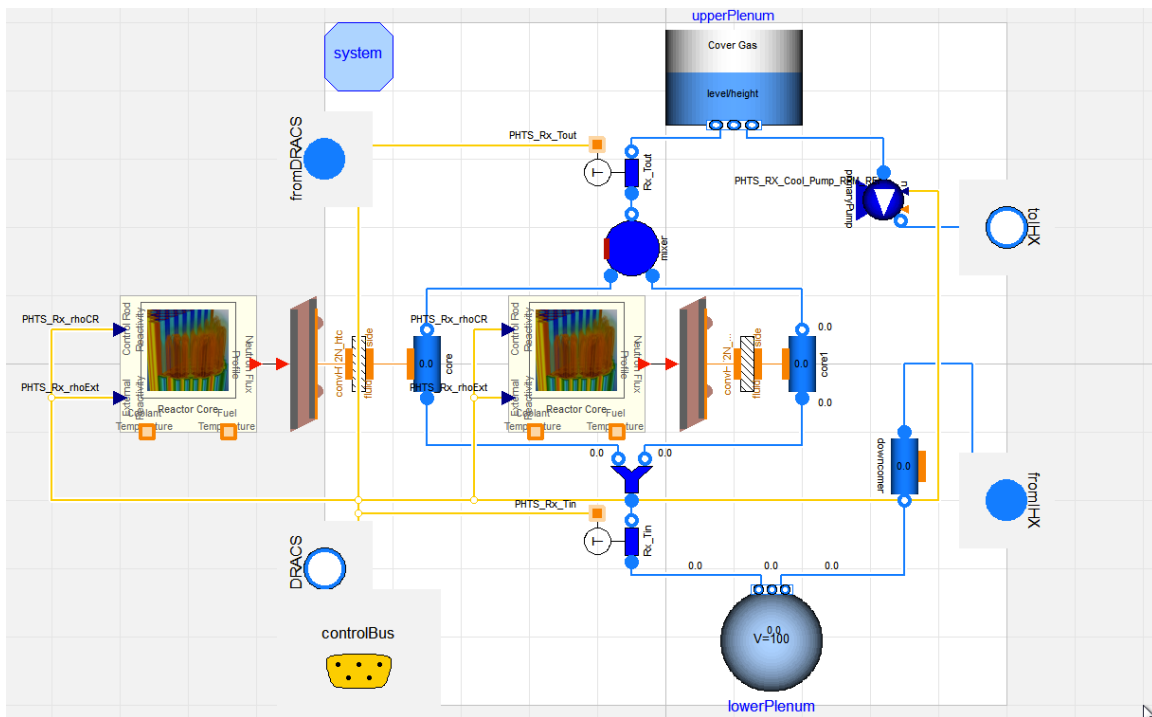


Fig. 34. Two-zone representation of the reactor core and connected fluid systems.

3.4 INTERMEDIATE HEAT EXCHANGER

The IHX for the molten salt architecture is structurally identical to the IHX for the ALMR previously developed (Fig. 35).

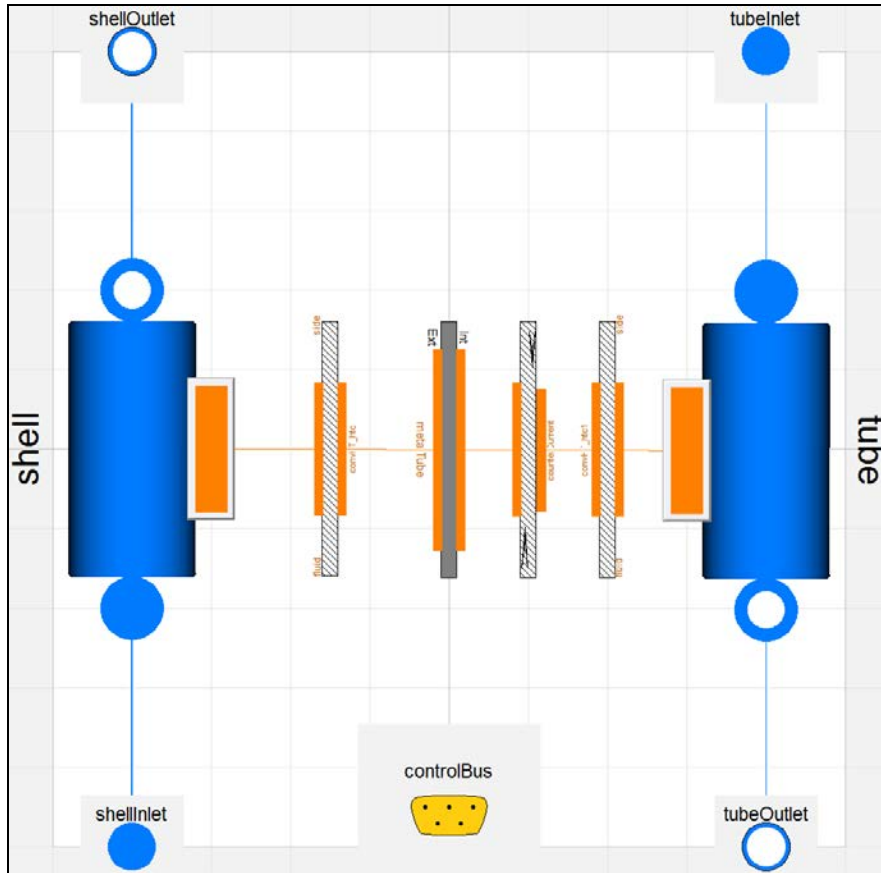


Fig. 35. SmAHTR IHX model.

Updating the parameters to represent the design of SmAHTR produces the following temperature profiles (Fig. 36), which are similar to the values given in the SmAHTR design documentation.

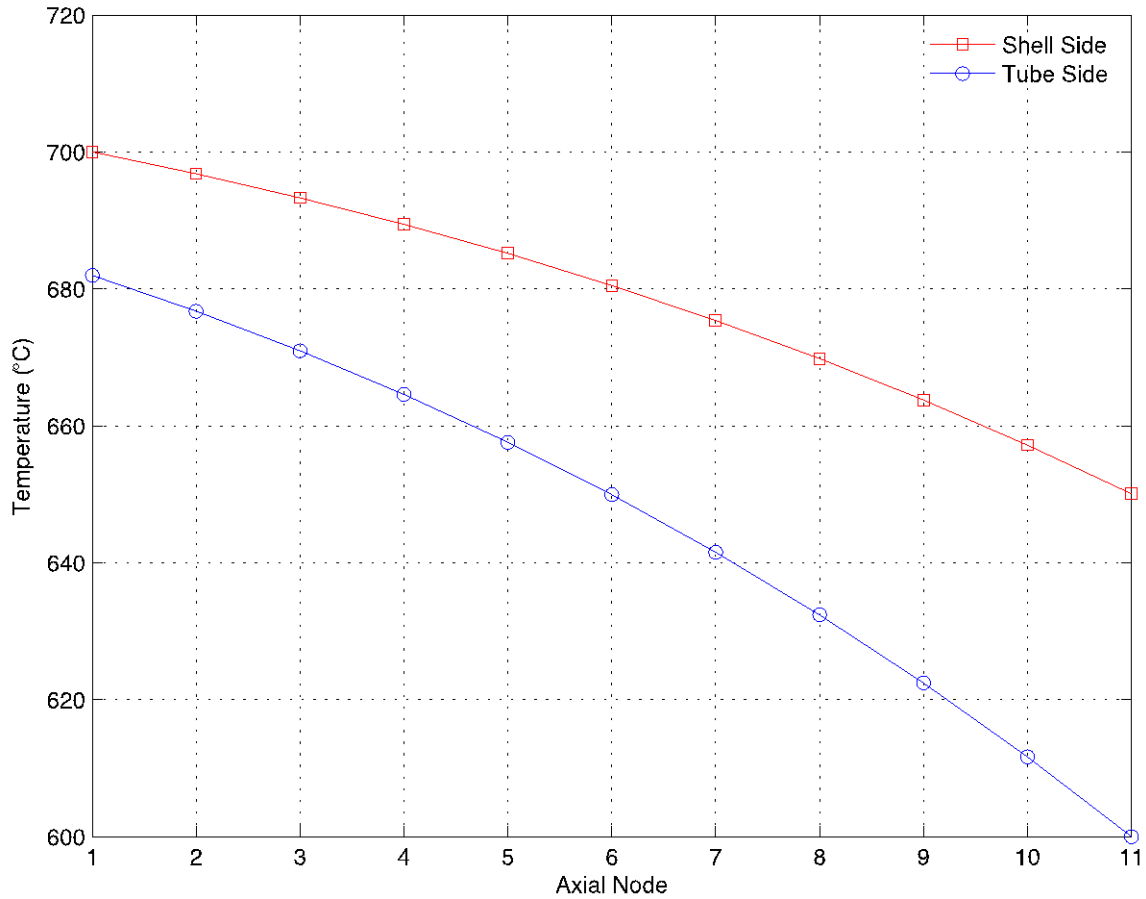


Fig. 36. Temperature profiles on the shell and the tube side as function of axial node.

3.5 INTERMEDIATE HEAT TRANSPORT SYSTEM

The intermediate heat transport model used in the SmAHTR to connect the IHX and steam generator is similar to the architecture implemented for AHTR (Fig. 37). It serves to account for the dynamic time delay between core power and steam generator heat removal.

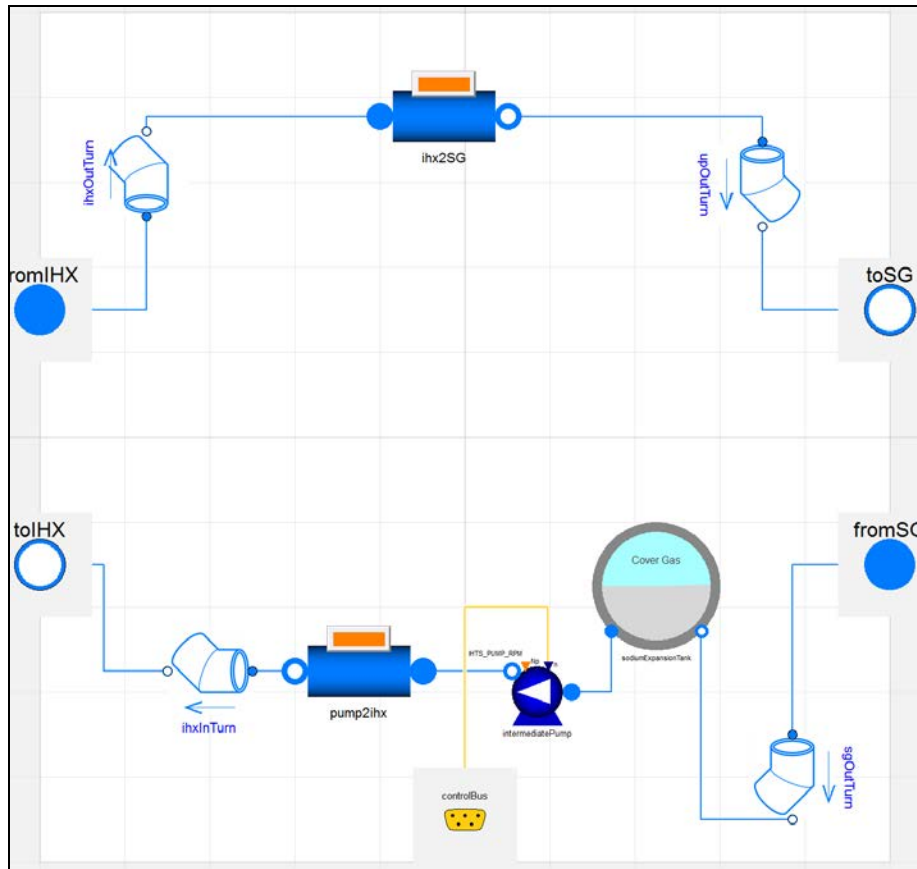


Fig. 37. Intermediate heat transport system.

3.6 STEAM GENERATOR MODEL

The steam generator model implemented in the FHR concept is similar to the steam generator model implemented in the ALMR (Fig. 38). Lacking design specifications for specific PCSs, the steam generator was designed to meet the known design characteristics of the SmAHTR, namely, a secondary flow rate of 3970 kg/s and a secondary temperature drop of 16.9°C in the steam generator. Improved model veracity was accomplished with the addition of two features. The shroud of the steam generator added thermal capacitance to the system by including a metal wall around the shell-side flow. The existing model for shell flow was modified to allow heat transfer to occur through internal and external annular heat ports, a capability that will be useful later in more accurately modeling the downcomer in the reactor vessel. Lower and upper plenums on the steam tubes were also incorporated, providing the mixing headers for the tube bundles.

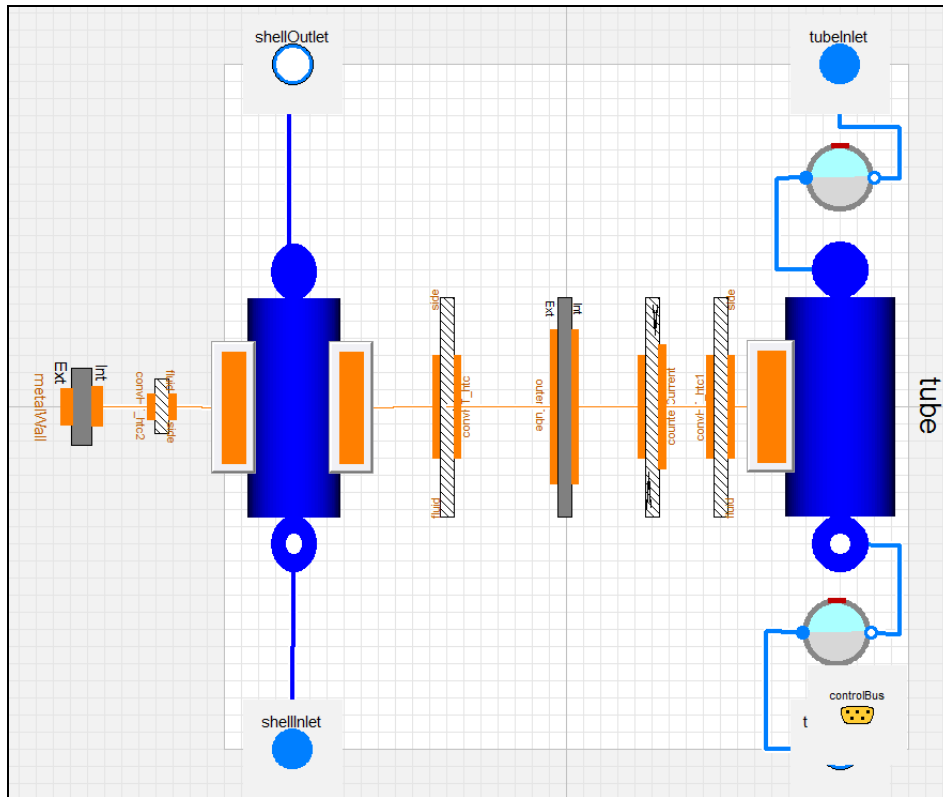


Fig. 38. Implementation of steam generator for FHR.

As a demonstration of the effect the plenums have on dynamics, a sawtooth wave function was input to the inlet enthalpy to simulate high-frequency dynamics (Fig. 39).

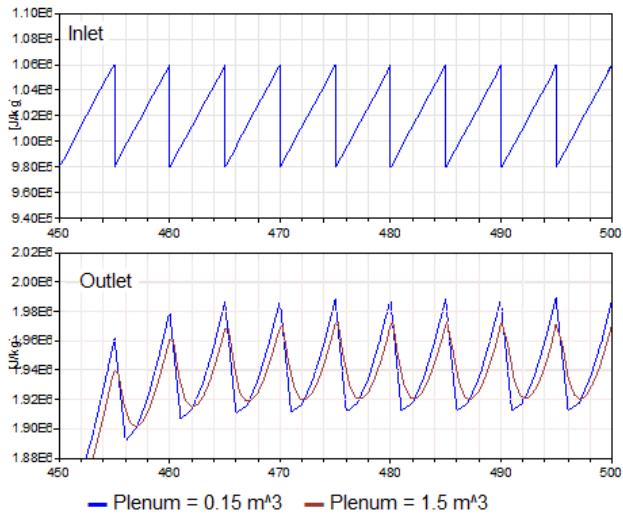


Fig. 39. Effect of plenums on steam generator dynamics.

Simulating steam generators using two different plenums demonstrates the effect of plenum size. The larger plenum, shown in red, provides capacitance to the system and acts as a low pass filter, screening out the high-frequency oscillations while adding a slight phase delay, making the system more realistic.

3.7 POWER CONVERSION SYSTEM

Literature available on the PCSs of the proposed SmaHTRs is quite limited. The flexibility of the reactor makes them useful for process heat, process steam, or electrical power conduction by myriad methods, such as supercritical steam, supercritical CO₂, and traditional Rankine cycles. The incorporation of these models into the ModSIM environment is ongoing and will be improved as more literature is generated on PCSs for small FHR concepts. Until then, to support primary heat transport system development, preliminary power conversion models are being generated to provide realistic approximations, without necessarily providing the optimized performance of systems to be incorporated at a later date.

3.7.1 Null Implementation

The Null implementation simply provides appropriate boundary conditions for other models without performing internal calculations. It is an extension of the PCS interface for the ALMR (Fig. 40).

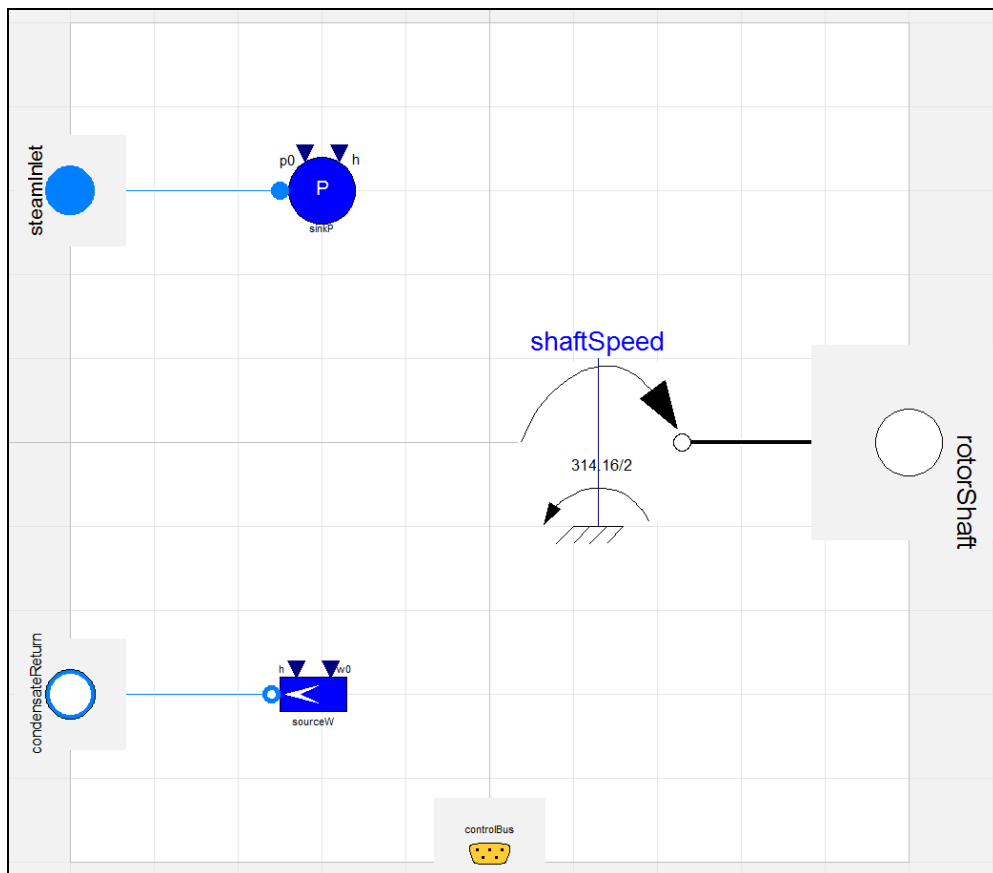


Fig. 40. PCS Null implementation.

The existing options for a PCS from the ALMR architecture have also been converted into the menu tree for FHR, with reasonable values substituted to match the energy imparted to the working fluid from the IHX.

As an improvement, the ability to separate steam from the IHX into saturated water and saturated vapor was undertaken, which would allow the steam cycle to be more accurately modeled. Instead of allowing a partial quality, saturated steam/water mixture to enter the turbine, the steam separator model splits the feedwater stream into two flows.

The PCS model with steam separation (Fig. 41) uses two turbines and open feedwater heaters, modeled as a mixing volume and an accumulator. It is currently still being improved to produce reliable convergence and dynamic response.

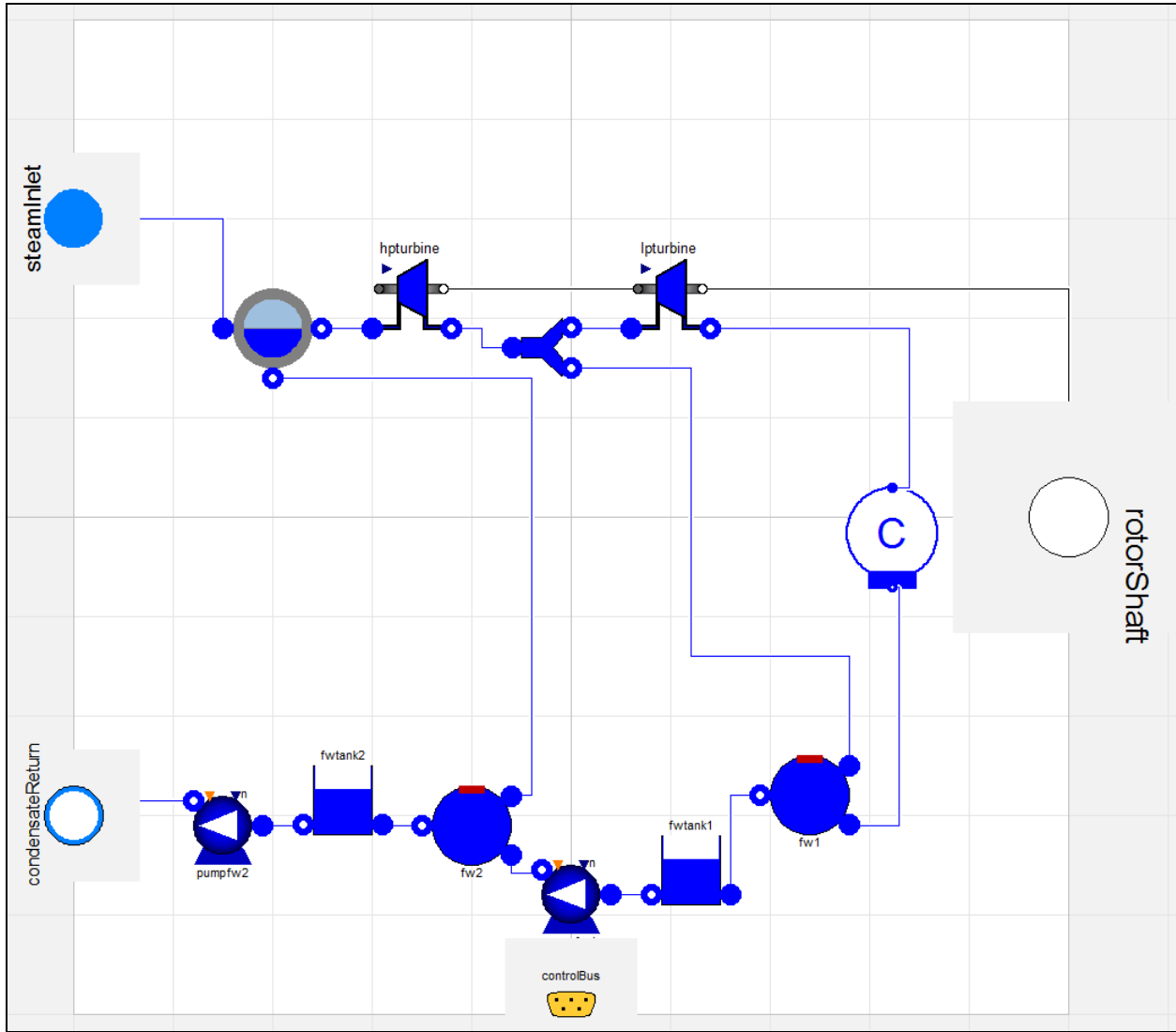


Fig. 41. FHR power conversion system with steam separation.

3.7.2 Steam Separator Model

In order to model Rankine cycles more accurately, a steam separator component model (Fig. 42) has been developed that separates a flow of mixed water and vapor into a flow of water and a flow of vapor. The model contains a continuity equation for mass and energy:

$$\begin{aligned} \text{der}(M) &= q_{\text{feed}} + q_{\text{vapor}} + q_{\text{condensate}} , \\ \text{der}(E) &= h_{\text{feed}}q_{\text{feed}} + h_{\text{vs}}q_{\text{vapor}} + h_{\text{ls}}q_{\text{condensate}} . \end{aligned}$$

Satisfying these equations while setting the enthalpies of the outflows fixed to those of saturated steam and water, respectively, governed the behavior of the model.

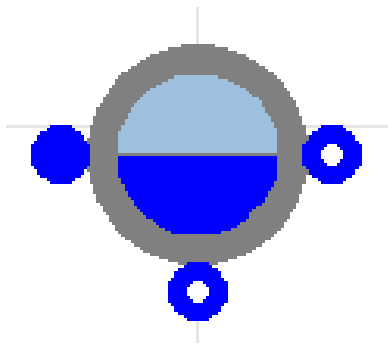


Fig. 42. Steam separator component.

3.8 ELECTRICAL GRID

For those applications in which power production is desired with the FHR model, 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. The electrical grid model is identical to that documented in Reference [1]. In the current implementation, the electrical generator is modeled as an ideal synchronous generator where the frequency in the electrical connector is the electromotive force (emf) 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 electrical grid is modeled with the swing equation, which calculates the load angle as a function of differences between the mechanical torque (provided by the turbine shaft) and the electrical torque (resistant torque generated by the electrical load). The load angle dictates the amount of power that can be transferred to the grid. The modeled power and frequency sensors monitor the load angle of the network. If the frequency gets out of range, an anticipatory automatic trip is initiated for the turbine and the generator within the model.

The objective of the project is to refine and couple these component and system models into an end to end system simulation that allows the exploration of various aspects of operation for advanced reactor concepts. The development and refinement of these component and system models (described here in Sect. 3) for various reactor architectures is a continuing part of the project task.

4. MOLTEN-SALT-COOLED I&C IMPLEMENTATIONS

4.1 TASK DESCRIPTION

The focus at this stage is the development of the FHR-cooled architecture in the MoDSIM environment. The previous primary system I&C implementations developed for the ALMR concept documented in Reference [1] are being implemented in the FHR-cooled architecture as well, including reactor control based on the core outlet temperature or the temperature difference. A comparison of the effect of a simple transient (reactivity) on the control aspects of each reactor type is the objective. In addition to this work, the development of additional reactor system control strategies for other plant systems and components is under way. This includes control for both the steam generator and the main steam turbine. A discussion of this ongoing work is provided in this section.

As documented in Reference [1], two control strategies (Figs. 43–44) have been developed to demonstrate reactor behavior for closed-loop power generation control.

Baseline “Null” Configuration With CS #1

Trapezoid Reactivity Transient, Reactor Rho Ext - 0.25 amplitude, 500 seconds in duration, @ time 2000 sec

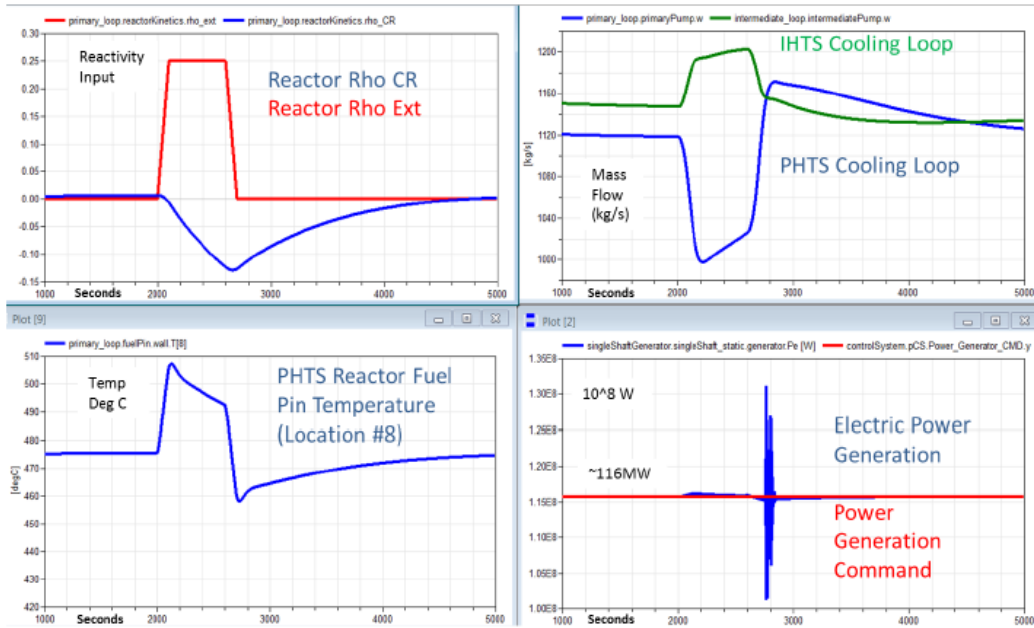


Fig. 43. Reactor I&C control strategy #1 (ALMR transient).

Baseline "Null" Configuration With CS #2

Trapezoid Reactivity Transient, Reactor Rho Ext - 0.25 amplitude, 500 seconds in duration, @ time 2000 sec

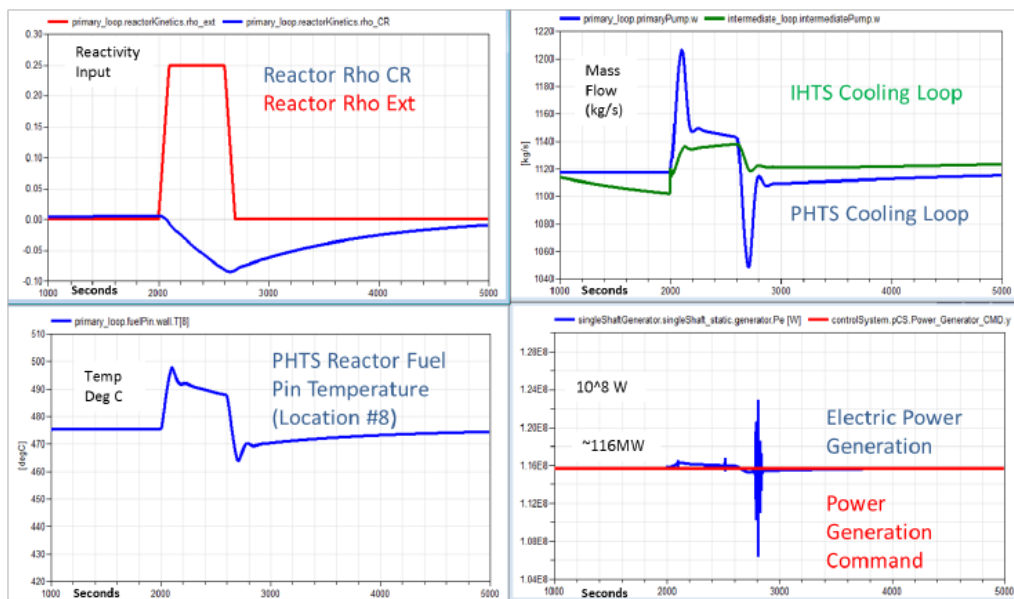


Fig. 44. Reactor I&C control strategy #2 (ALMR transient).

Figure 45 illustrates the selection of model classes for the control strategy #1 configuration. One considerable advantage in developing a consistent modeling framework is that this approach allows for different architectures to be implemented in a straightforward fashion. Therefore, although the I&C overlay for the FHR architecture has not yet been completed, the implementation will be identical to that shown in Fig. 45 except for different dropdown menu selection choices. Similarly, the initiation of events through the salt-cooled architecture will be implemented through drop down choices in the menu as displayed in Fig. 45.

Currently, the I&C implementations have not been fully incorporated into the FHR architecture. With this new architecture complete, and following the implementation of the I&C control strategies for reactor control, the next goal will be to make a direct comparison of control strategies for ALMR and FHR architectures. At that time an assessment of the reactor control strategies will be presented for both architectures with a discussion included of the implications to other advanced reactor design concepts. This work is expected to be complete in the next project deliverable scheduled for December 2014.

Baseline "Null" Configuration with Control Strategy #1
 ORNL_AdvSMR.PlantSystems.Rx2Grid2

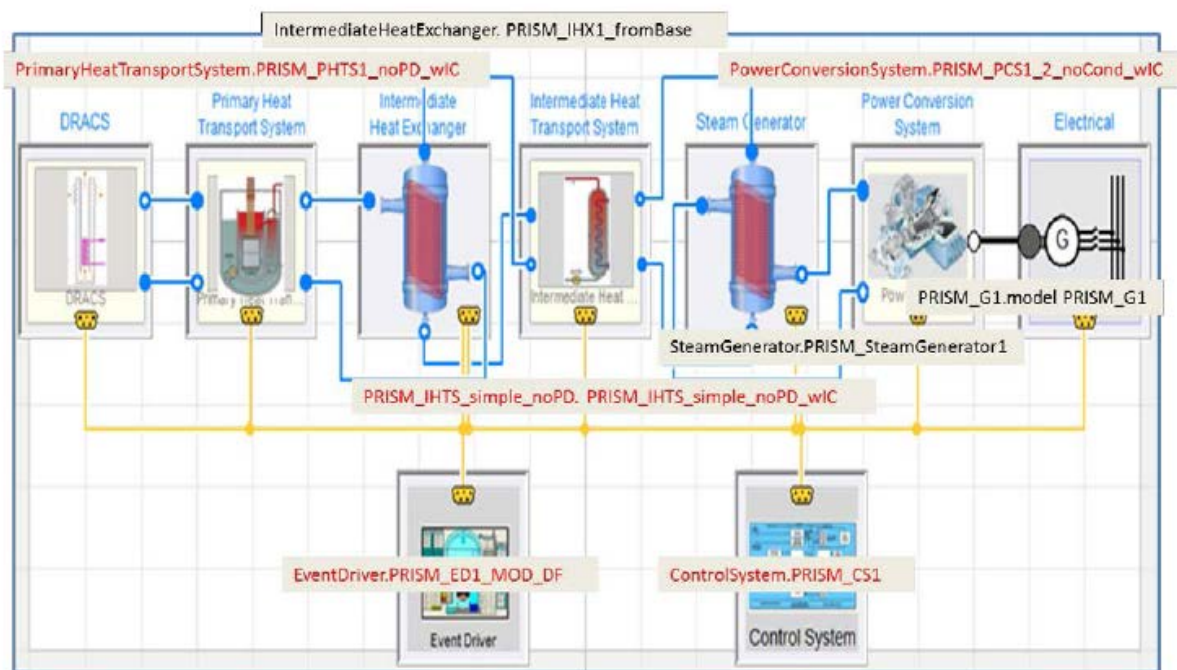


Fig. 45. Architecture selection I&C control strategy #1 (ALMR model).

4.2 ADDITIONAL I&C CONTROL IMPLEMENTATIONS

In addition to reactor control, the overall system architecture must include other control systems to handle the integration of multiple systems across the plant. Current plans include the addition of a steam generator control loop as well as a control loop for the main steam turbine. Ongoing work in these areas is described in Sects. 4.2.1 and 4.2.2.

4.2.1 Steam Generator Control Implementations

The steam generator outflows steam to the turbine for power generation. Makeup water must be supplied by the inflow of feedwater to maintain the proper balance of water and steam in the steam generator. A mismatch in the steam outflow and the feedwater inflow will cause a change in the steam generator drum level. The feedwater inflow is typically regulated using a drum level control loop. In the steam drum level control loop, the level is the only measured parameter. The feedwater inflow is typically regulated using a control valve and verified by a flow control measurement (Fig. 46).

The baseline model will be updated to include control using the feedwater system with the appropriate pump, flow loop, control valve, and flow measurement. The model will also be updated to include the steam generator drum characteristics with a drum level measurement. A control loop will be added to provide the steam generator drum level control for power plant steady-state and dynamic operation.

The behavior of the steam generator drum for different power generation levels and conditions can be determined from steady-state and dynamic conditions. The storage capability of the steam generator drum system can be examined for power generation dynamics or system failures.

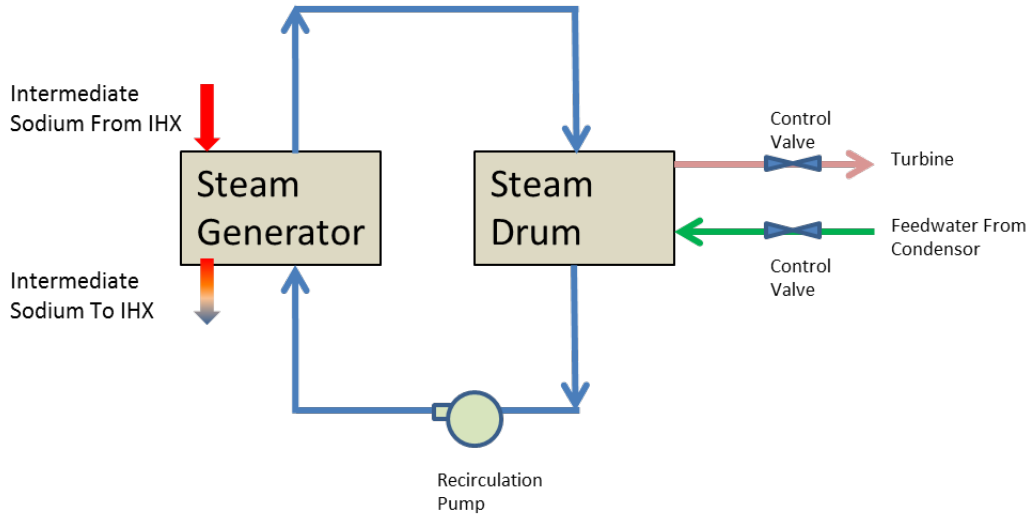


Fig. 46. Steam generator functional diagram.

4.2.2 Main Steam Turbine Control Implementation

In modern power plants, a throttling or governing steam valve is used to adjust the steam pressure and flow supplied to the turbines to maintain a constant speed at varying generation loads. This valve is used to either regulate the direct supply flow or a bypass flow path to maintain the desired turbine speed. Figure 47 illustrates the relationship of valve area to steam pressure and flow, which ultimately affects the generator speed and the electrical frequency. It should be noted that this approach is not used to vary the actual power output, which is performed by controlling the energy provided to the turbine by the reactor and the heat transport systems.

The baseline liquid metal reactor model presented in Reference [1] was examined for the addition of a throttle valve to the high pressure turbine steam supply in the PCS. Figure 48 illustrates the updated model with the throttle valve and the associated control concept. The throttle valve parameters are shown in Fig. 49. These parameters were selected based on PRISM Reference [7] but have not been formerly validated.

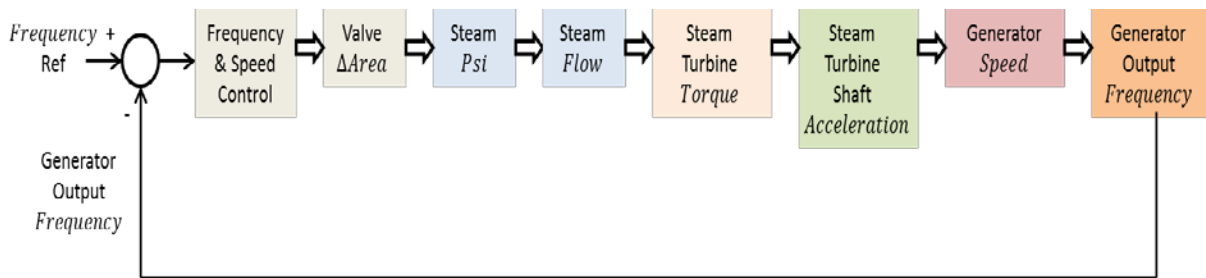


Fig. 47. Turbine speed control concept diagram.

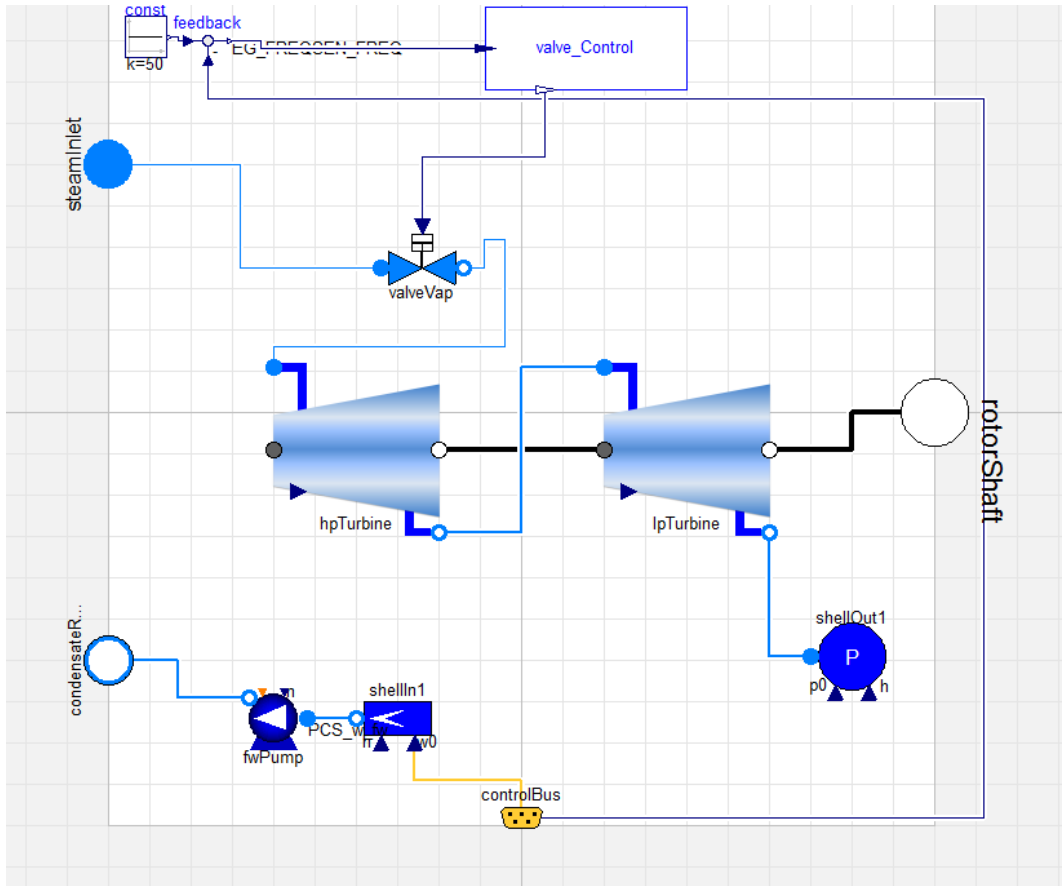


Fig. 48. Baseline model PCS with throttle valve added to the high-pressure turbine supply.

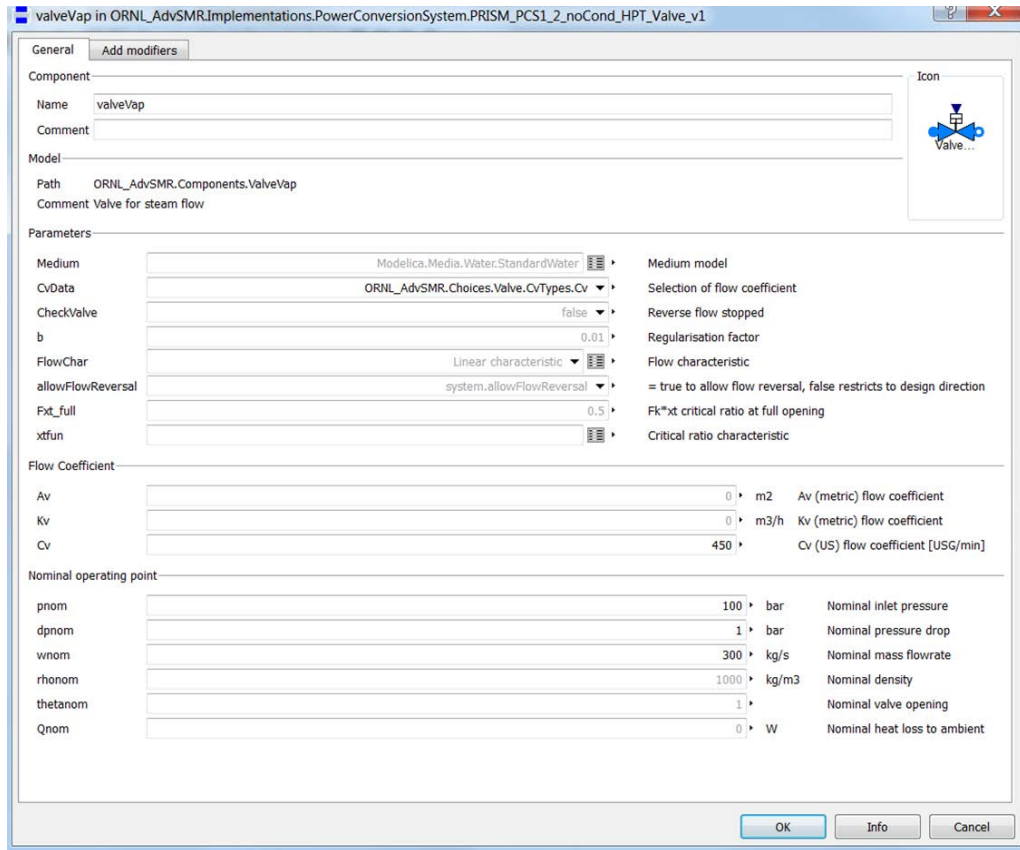
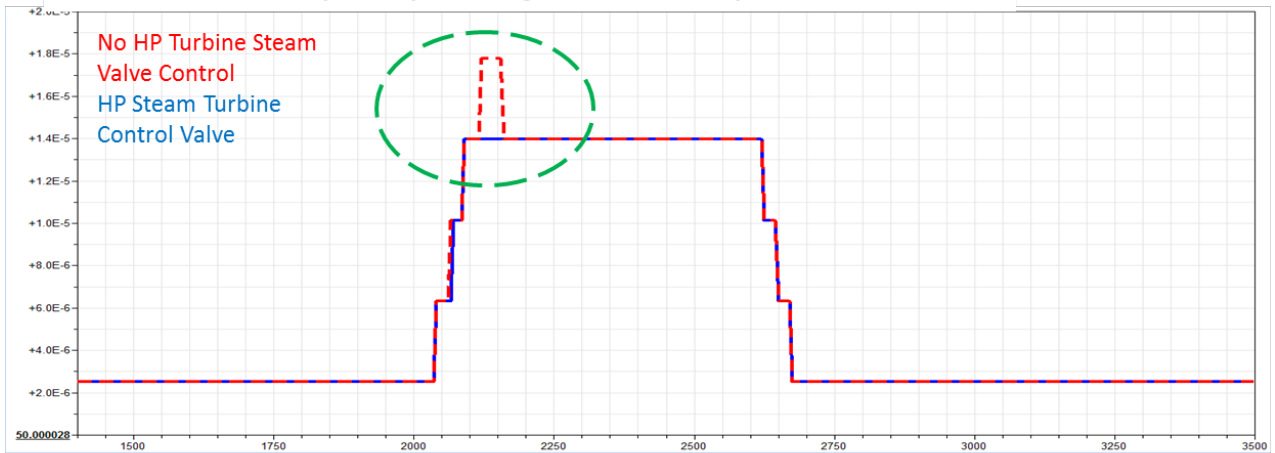


Fig. 49. Throttle valve parameters.

The valve control method is a proportional and derivative-based approach that compares the actual generation electrical frequency with the desired reference and adjusts the valve opening based on the frequency error. A derivative function is used in order to react quickly in a predictive manner. The derivative acts due to the rate of change of the error, which is sensitive to quick changes in the generator frequency due to a power plant dynamic change. Figure 50 illustrates that the throttle valve dynamically closing ~20% can reduce the generation frequency error during a power plant dynamic test of shifting power ~36% from 111 MW to 151 MW. The peak frequency error was reduced ~28% with the addition of the throttle valve control. The actual values of frequency error and the dynamic behavior should be considered only as representative because the turbine and generator inertia values are conceptual. Figure 51 illustrates the throttle valve effect on pressure and flow for the steam turbine. The pressure is increased ~14% which increases the enthalpy flow ~4%.

Generator Frequency During Reactor Output Test Profile



HP Turbine Steam Valve Area During Reactor Output Test Profile

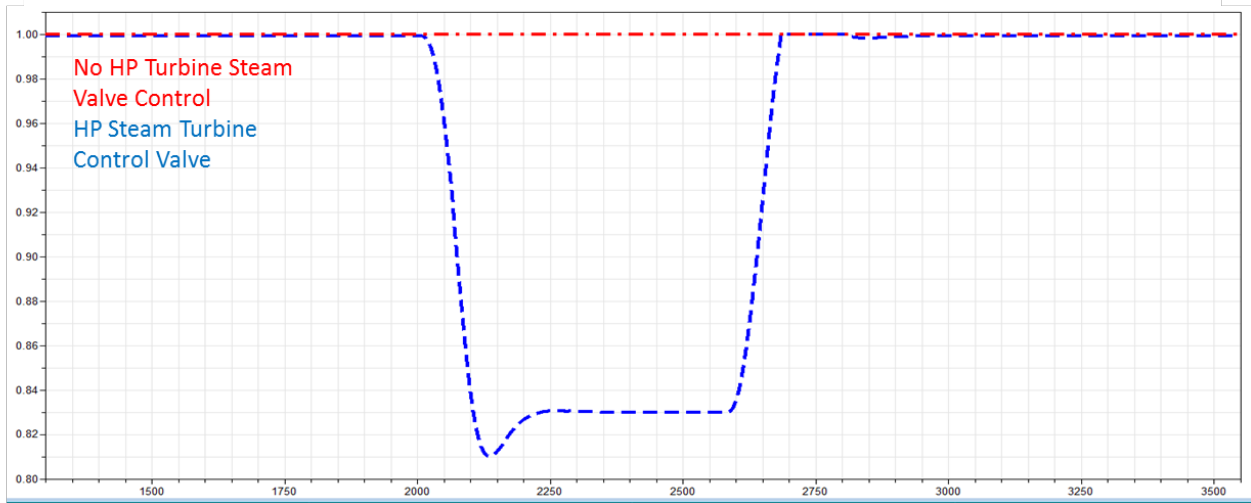


Fig. 50. Power plant dynamic test shifting power from 111 MW to 151 MW with high-pressure steam turbine throttle valve control.

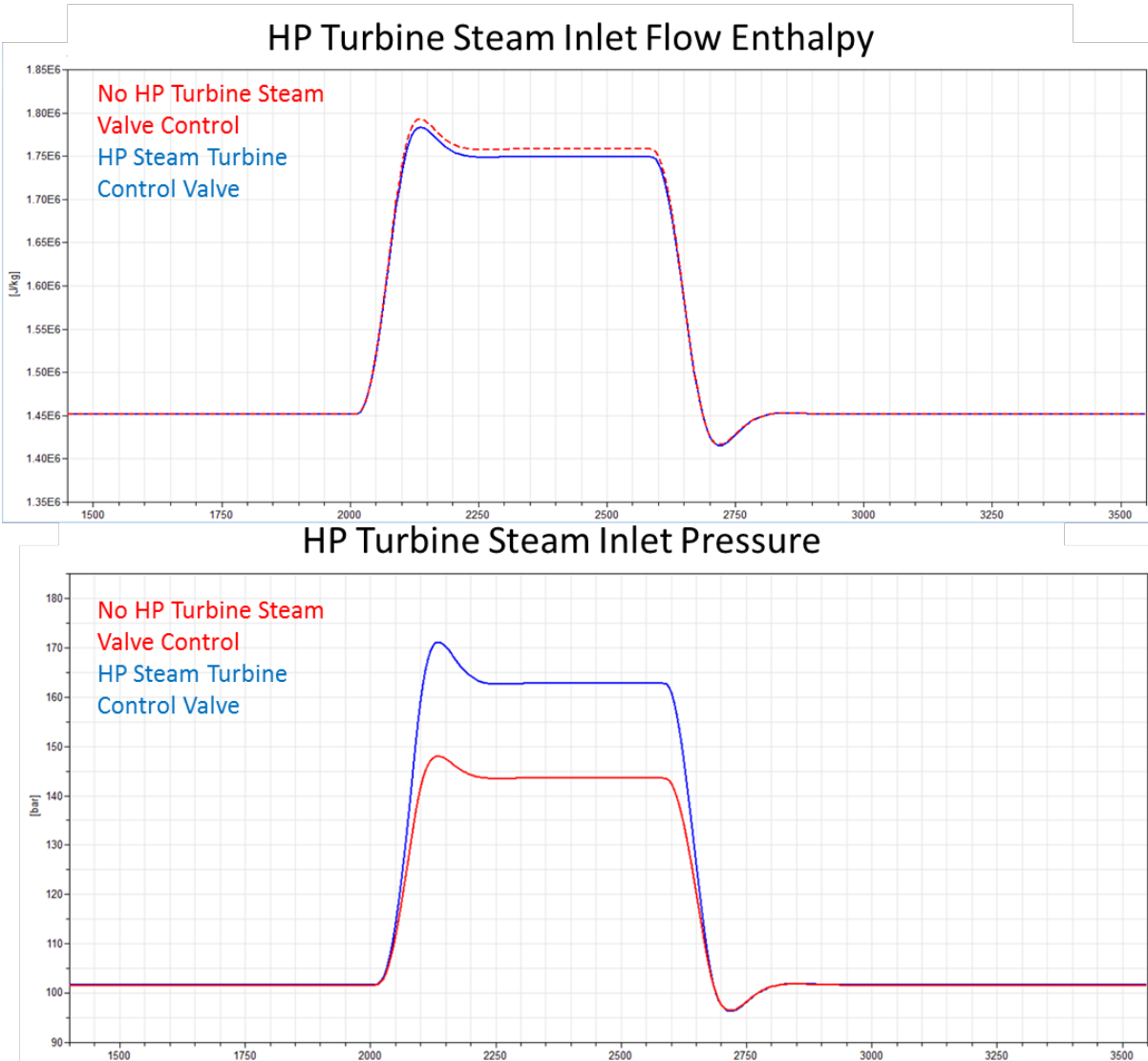


Fig. 51. Power plant dynamic test shifting power from 111 MW to 151 MW with high-pressure steam turbine throttle valve control.

5. COLLABORATION IMPLEMENTATION

5.1 TASK DESCRIPTION

The ultimate goal of this project is not only to develop different advanced SMR design models but also to promote collaboration among multiple designers with various levels of modeling experience. Several aspects necessary to achieve this goal are discussed in this section. The number and nature of repositories for the models are discussed, along with the initial partners, as well as simplified instructions for registering and downloading and installing the collaboration software. Finally, a brief update on the web-based application is provided. This application is scheduled for beta release as part of the December 2014 deliverable.

5.2 MODEL REPOSITORIES

All the initial Modelica models for this project are currently archived in two repositories. One is administered by Xogeny, and a second is administered by ORNL. Xogeny used the ALMR models to develop the initial prototype of the web application for the ALMR reactor concept. All current FHR-related models are contained in an ORNL repository. The project will eventually keep all models exclusively in an ORNL repository for collaboration work, and all models will be transitioned accordingly. To facilitate this transition, however, consideration must be given to the various levels of collaboration and verification expected to be required for a wide range of potential proprietary and non-proprietary partnerships. This may require multiple repositories with different access permissions.

Multiple models are expected to be developed simultaneously as part of this project. These models will include multiple reactor designs, with various systems and components. To accommodate this, several different repositories may be needed. Consistent with many software development projects, software branches within a repository can be established in a tiered hierarchy that allows development of the models to progress based upon different levels of quality assurance/verification. Each of these tiered branches is discussed in Appendix A. Currently, a single repository called “OpenCollaboration” has been established. Other repositories will be created if proprietary work or other necessary workflows dictate. Beyond the repository, the management of workflows is necessary to establish a pedigree for different modeling efforts. These are handled by “forking” branches off the main repository. The anticipated branches provide for various expected quality assurance (QA) levels as models work through the system from initial implementation to final production. These branches are also discussed below. As noted above, it is anticipated that some models in the future may be proprietary. For these models, a separate repository with parallel “Development,” “QA,” and “Production,” branches will be provided and will be created to suit the needs of potential customers as they arise. The strategy for these repositories and branches is discussed in more detail in Appendix A.

5.3 COLLABORATION PARTNERS/PROTOCOL

As discussed in Reference [1], the software platform for collaboration chosen for this project is GitHub. The basic details for GitHub and the rationale for its selection are discussed in Reference [1]. At the highest level, GitHub is an open web site that utilizes the “Git” protocols for file sharing/modifying and version control to coordinate software modifications. After registering on GitHub and installing TortoiseGit, the collaborating user will request access to the appropriate ORNL SMR modeling repositories. Initially this will be the OpenCollaboration repository until other models are generated. The user will “clone” the OpenCollaboration repository on his/her local machine. This will allow changes in the local and master repositories to be pushed and pulled, and the repositories to be synchronized together. The use of GitHub will allow configuration control, with each change in the repository identified and documented. Reversion to previous repository versions is possible under this configuration control. The setup for the user’s local machine and the GitHub account are discussed along with the protocols and initial steps required to modify and share a model in Appendix A.

5.3.1 Collaboration Partners

Collaboration in this project is an ongoing exercise. To-date, initial collaboration has been with business partners through the development of the necessary tools required to fulfill the project’s goals and objectives. As documented in Reference [1], these business partnerships include Modelon Inc. and Xogeny Corporation. Modelon provides the FMI application, and Xogeny has developed the initial web application. Other partnerships are being added as well- These partnerships include both universities and

other DOE laboratories that are engaged in nuclear reactor development, as well as business partners that help develop the tool capabilities. ORNL is also engaged in establishing partnerships with industry that will help facilitate greater interest in utilizing the Modelica modeling paradigm for nuclear-related applications. Beyond the aforementioned business partnerships with Modelon Inc. and Xogeny corporation, the partnerships identified below are in the early stages of development. Discussions are being conducted to identify tasks that will complement each partner's areas of focus and responsibilities.

University Partnerships

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

Business Partnerships

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

5.3.2 Web-Based Collaboration

The previous section highlighted collaboration within model development. A second collaboration potential exists in the generation of model results. As discussed in Reference [1], the web-based application being developed allows users of various skill levels to generate results from a library of compiled models. During this period of performance, no additional implementations for the web-based application were created. The next deliverable will include the extension of the web application to include the FHR concept (SmAHTR) along with the ALMR implementation that currently exists in the web application. In addition to including this second reactor architecture type, the web-based application will be finalized for a first beta release that will be made available to the DOE sponsor as well as the collaboration partners.

6. INTEGRATING ACTIVITIES

The MoDSIM tool has been designed to provide a convenient way to develop and collaborate on advanced reactor design models. To this end, one key measure of success is the delivery of models to other DOE advanced reactor programs and projects for their use.

6.1 REMAINING FY 2014 TASKS

The focus of this deliverable is to meet one of the key FY 2014 goals, that is, to extend the library models to a second architecture. The FHR architecture and models described in this report fulfill this objective. The last remaining FY 2014 task is the expansion of web-based collaboration and the establishment and release of the initial DOE access tool. In addition to these stated activities, the transfer and application of this tool into other projects under the AdvSMR R&D program are required to ensure the tool becomes a vital part of the design community. Progress towards the integration of this tool into each of these projects is discussed in the following sections.

6.2 ADVANCED REACTORS I&C INTEGRATION ACTIVITIES

6.2.1 Supervisory Control

The supervisory control necessary for a multi-modular SMR plant project is an area of ongoing research. This project is proceeding with the development and demonstration of an architectural framework and foundational modules that are needed to facilitate the integration of control, decision, and diagnostics to support the necessary level of automation required for multi-modular supervisory control system. Properly implemented, the innovative use of intelligent automation—via a supervisory control system—can have a significant impact on optimizing plant staffing and controlling operating and maintenance costs. Essentially, the economy of automation can serve as a compensating factor for the loss of economy of scale.

Traditionally, dynamic systems have been modeled in continuous-time modeling environments. However, in order to simulate the supervisory control system execution, the simulation environment must support both continuous-time evolutions and discrete-event evolutions. Discrete time refers to a sampled representation of a continuous-time system, while discrete events are instantaneous transitions upon a certain trigger. An example would be opening or closing of a pump or trip of a turbine. The supervisory control project team is working with the Advanced SMR Dynamic Modeling Tools project to develop system and component definitions with proper interfaces, which are the communication channels that would transmit the supervisory controller commands to the functional layer, and similarly, receive sensor readings and fault indications to execute decision-making strategies.

The primary objective of the supervisory control system is to increase the level of automation and to reduce the cognitive load on reactor operators by taking on routine operator actions executed primarily during normal operations, and some actions performed during startup and shutdown. In addition to routine operator actions, the supervisory control system will intervene during off-normal conditions such as component failures or unexpected transients. The supervisory control system is not intended to replace the operator as the key decision node for safety-related actions, nor is it to support or complement protective actions performed by reactor protection or engineered safety features actuation systems.

Minimizing the likelihood of a trip in individual reactor modules is one of the key tenets of supervisory control. Shutdown of a reactor unit has serious economic repercussions. The output of the reactor is lost and, typically, an unplanned or unanticipated trip requires that a series of tests—generally required by the regulator—be performed following the reactor trip. Furthermore, if the trip occurs later in the cycle, the

core may not have sufficient excess reactivity for immediate restart due to xenon buildup in thermal reactors, which may require time for xenon levels to drop below the threshold of excess reactivity.

The supervisory control system uses techniques such as state estimation, diagnostics, prognostics, probability assessments, physics-based assessments, and procedural-based assessments to determine viable decision options that can be used with mathematical operations to select a single prime candidate decision option. This prime candidate decision option is then verified with dynamic models and analysis to determine whether the decision option meets the intended outcome before the action is directed to the execution levels. In Fig. 52, a verification feedback path is used to refine the determination of the decision options step.

The supervisory control system project leveraged the end-to-end dynamic simulation capability developed under this project. This capability is being used in deterministic portion of decision-making (Decision Analysis box) and for verification of actuation (Verification box) in Fig. 52.

As part of a future deliverable, the supervisory control project will make use of one of the developed MoDSIM SMR simulation models for the evaluation of supervisory control concepts. Currently the end-to-end ALMR and FHR models exist for this application.

6.2.2 Advanced PRA

The advanced PRA project is related to the supervisory control project. The objective is for probabilistic models developed with the advanced PRA along with deterministic models developed in the SMR Modeling project to supply input to the supervisory control project. This will be accomplished by the integration of advanced physics modeling, probabilistic analysis, and computational methods. The advanced PRA project will make use of a MoDSIM SMR simulation model for the evaluation of advanced PRA concepts as part of its future activities.

6.2.3 Hybrid Energy Systems

Hybrid energy systems are an active area of energy research. The potential combination of reactor-based energy systems coupled in nontraditional ways has given rise to an optimization challenge for future energy and industrial use practices. Discussions with personnel working in this area at Idaho National Laboratory have identified a strong customer need within this community for flexible system modeling tools such as the toolset developed under this project, and a collaboration has been established.

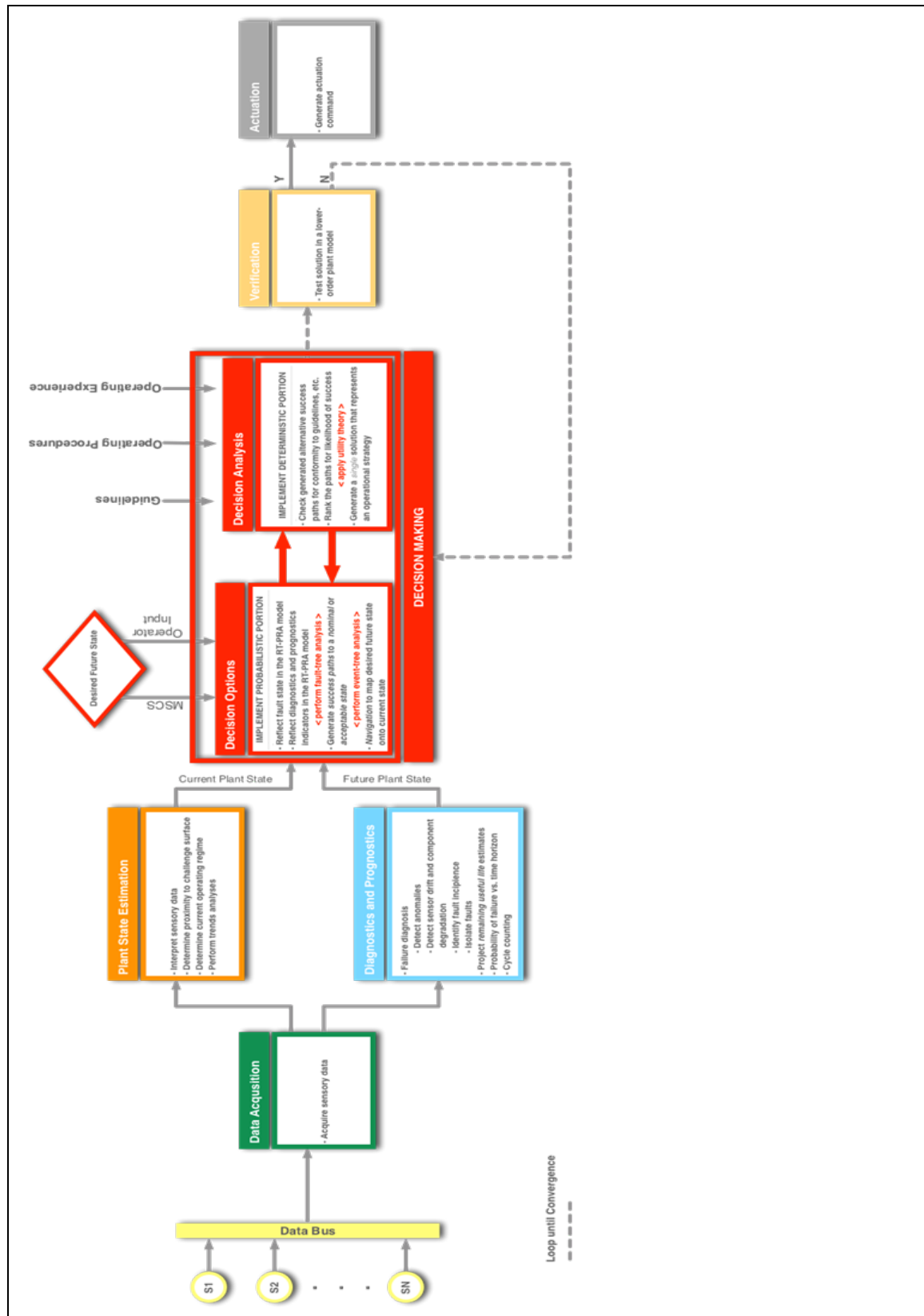


Fig. 52. Architecture for local supervisory control system and functional modules of decision-making.

7. REFERENCES

- [1] *Update on Small Modular Reactors Dynamic System Modeling Tool*, ORNL/TM-2014/50, Oak Ridge National Laboratory, Oak Ridge, TN, March 2014.
- [2] *Preconceptual Design of a Fluoride-Salt-Cooled Small Modular Advanced High-Temperature Reactor (SmAHTR)*, ORNL/TM-2010/199, Oak Ridge National Laboratory, Oak Ridge, TN.
- [3] C. Folsom, C. Xing, C. Jensen, H. Ban, and D. Marshall, “TRISO Fuel Thermal Conductivity Measurements,” *Trans. American Nuclear Society*, v. 107, San Diego, CA, Nov. 11–15, 2012.
- [4] [4] Haynes International, Hastelloy N Alloy Fact Sheet, <http://www.haynesintl.com/pdf/h2052.pdf>
Haynes International, Hastelloy N Alloy Fact Sheet, <http://www.haynesintl.com/pdf/h2052.pdf>
- [5] P. Avigni and B. Petrovic, “Fuel Element and Full Core Thermal-Hydraulic Analysis of the AHTR for the evaluation of the LOFC Transient,” *Annals of Nuclear Energy*, v. 46, pp. 499–510 (2014).
- [6] D. T. Ingersoll, et al., “Status of Preconceptual Design of the Advanced High-Temperature Reactor (AHTR),” ORNL/TM-2004/104, Oak Ridge National Laboratory, Oak Ridge, TN (2004).
- [7] PRISM Preliminary Safety Information Document, GEFR-00793, UC-87TA, December 1987.

APPENDIX A. INTRODUCTORY COLLABORATION MATERIAL

APPENDIX A. COLLABORATION MATERIAL DETAIL

As a brief background the project work description and high level objectives are included from Reference[1]:

Project Work Scope

“...In particular, the safety and control evaluation of the possible concepts depend on an understanding to the system dynamics necessitating a number of mathematical models of the plants. A great many different organizations and researchers may be involved in evaluating the concepts. To facilitate this collaboration the project was instituted with the following goals and objectives. The goal of the project is to facilitate rapid development of SMR models, ensure consistency among research products while minimizing duplication of effort.”

Project Objectives

The high level objectives include the following:

- (1) Define a standardized, common simulation environment that can be applied throughout the program.
- (2) Develop a library of baseline component modules that can be assembled into full plant models.
- (3) Define modeling conventions for interconnecting component models, and
- (4) Establish interfaces and support tools for users of various capabilities to facilitate simulation development (i.e., configuration and parameterization), execution, and results display and capture.

ORNL SMR Modeling Background Material

As an introduction to the project, some initial slides have been prepared that explain the purpose, objectives, and goals of the project, along with the necessary first steps to begin collaboration. These slides are presented in Figs. A.1–A.13.

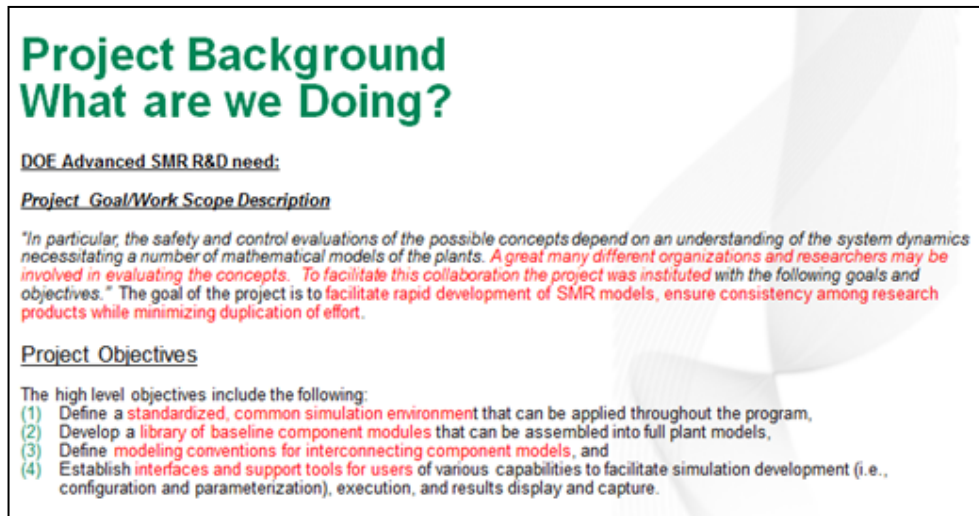
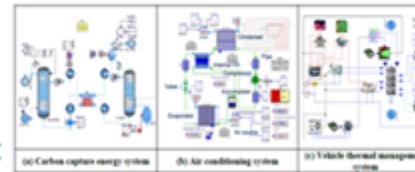
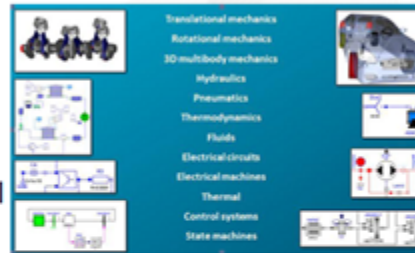


Fig. A.1. ORNL SMR background presentation – Slide 1.

Why Modelica (in general)?

- Modelica is a nonproprietary, object-oriented, equation-based language used to conveniently model complex physical and cyber-physical systems.
- Modelica is a multi-domain modeling language and can handle mechanical, electrical, electronic, fluid, thermal, hydraulic, pneumatic, magnetic, and control systems and their interactions
- **Been adopted by other industries for fast turnaround design studies (automotive/aerospace)**
- In short, its free, open source, object oriented and has multi-domain libraries that already exist and can easily be modified

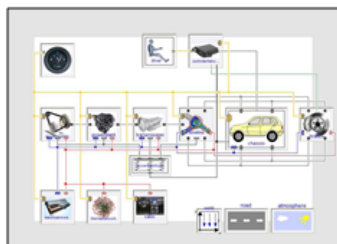
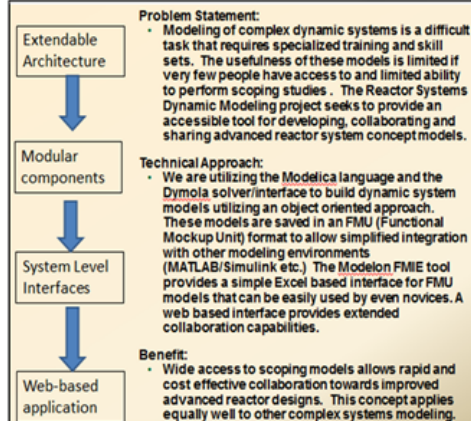


"Efficiency in the model development and accuracy in the modeling results can be achieved by developing re-useable, verified component models in library from which a system model can be readily assembled for a specific analysis."

Fig. A.2. ORNL SMR background presentation – Slide 2.

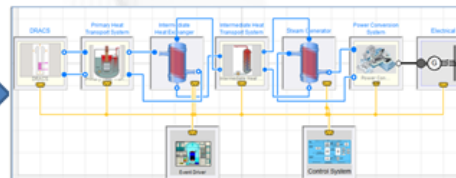
Why Modelica (for SMRs)?

- Model-based Modelica systems development for commercial (FORD) and military (DARPA) vehicles **has proven effective in reducing development time** and increasing product quality and reliability.
- A similar approach can be adopted for design optimization and analysis of key systems in support of advanced reactor concepts.



Modelica System model for automotive vehicle

Tech Transfer



Modelica System Model for Advanced Reactors

Fig. A.3. ORNL SMR background presentation–Slide 3.

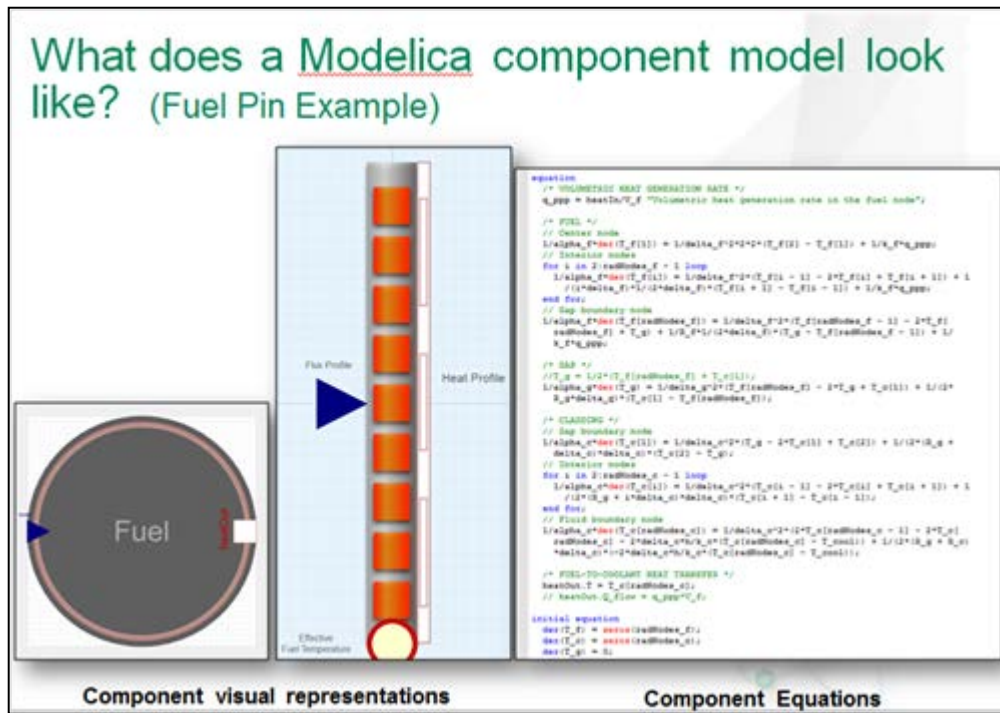


Fig. A.4. ORNL SMR background presentation–Slide 4.

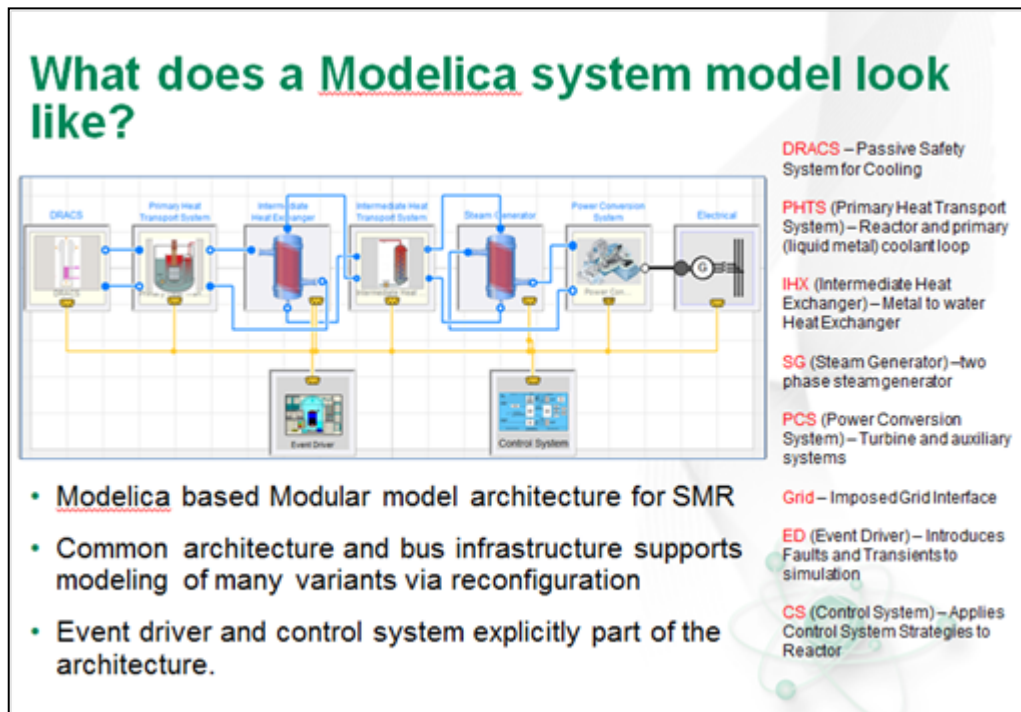
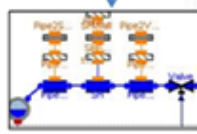


Fig. A.5. ORNL SMR background presentation–Slide 5.

How do we solve and access these models?

- We are modeling complex advanced reactors in the MODELICA language
- We are solving MODELICA models with DYMOLA solver
- We are leveraging the MODELICA ThermoPower library for reactor systems



Modelica is an object-oriented declarative, multi-domain modeling language for component-oriented modeling of complex systems, e.g., systems containing mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents.

Dymola is a commercial modeling and simulation environment based on the open Modelica modeling language. Dymola has unique multi-engineering capabilities which means that models can consist of components from many engineering domains/libraries that exist.

The ThermoPower library is an open-source Modelica library for the dynamic modeling of thermal power plants and energy conversion systems. It provides basic components for system-level modeling, in particular for the study of control systems in traditional and innovative plants.

BUT... The Game Changer is, can we help others to do this without in-depth knowledge of these tools or access to Dymola?

Yes, But there were three missing pieces

Fig. A.6. ORNL SMR background presentation–Slide 6.

The First Missing Piece... Modifying and Executing Models outside of Dymola



– FMI Excel Tool Allows sophisticated compiled models to be modified and executed through an Excel Interface, without a Dymola License

Functional Mockup Interface Add-in for Microsoft Excel® enables steady-state and dynamic simulation of physical models integrated in a spread sheet environment. Batch-simulations and parameter sweeps are accessible through easy to use functions that enable users to quickly get up to speed with sophisticated model analyses

Meta Data				Default Case 1 Case 2 Case 3			
Sheet version	Generated by Modelica FMI Add-in for Excel version 1.0						
Model name	PowerPlantControlTest_2dc						
Generation tool	Dymola Version 2017(22-Nov-2017)03.20						
FMI kind	CoSimulation_Standard						
Settings							
!start				10			
!stop				10	100	100	100
!FMI				C:\Users\rd\Documents\ControlTest\PowerPlantControlTest_2dc.fmu			
log level				INFO			
enabled				TRUE			
output points				100	100	100	100
Indata							
Name	Variability	Type	Unit				
steamPlant_Sim_1NRSQ.dums.HP4.Tmstart	parameter	Real	K	300			
steamPlant_Sim_1SourceGas.v0	parameter	Real	kg/s	535.5			
Watt_Gask	parameter	Real	W	5.4E+06			
ramp.height	parameter	Real		10			
ramp.duration	parameter	Real	s	50			
ramp.offset	parameter	Real		100			
ramp.startTime	parameter	Real	s	500			
P1.L	parameter	Real	1	250	250	25	10
P1.T	parameter	Real		100	100	100	200
ramp1.height	parameter	Real		10			
ramp1.duration	parameter	Real	s	10			
ramp1.offset	parameter	Real		0			
ramp1.startTime	parameter	Real	s	700			
Outdata							
Name	Variability	Type	Unit				
steamPlant_Sim_1NRSQ.GasOut.v	continuous	Real	kg/s	591	590	587	
steamPlant_Sim_1NRSQ.HeatExchangersGroup.Ec2_HP.water	continuous	Real	kg/s	63.2	63.64	63.77	
steamPlant_Sim_1NRSQ.HeatExchangersGroup.Ec9BP_EcP.g	continuous	Real	kg/s	590.6	590.3	587.3	
steamPlant_Sim_1NRSQ.HeatExchangersGroup.Ec_LP.gasOut	continuous	Real	kg/s	591	590	587	
steamPlant_Sim_1NRSQ.HeatExchangersGroup.Ec_LP.waterO	continuous	Real	kg/s	-95	-85.7	-85.2	
steamPlant_Sim_1NRSQ.HeatExchangersGroup.mHP.T	continuous	Real	K	631.7	631.7	632	
steamPlant_Sim_1sTG_3LPb.steamTurbines.valveP.Tm	continuous	Real	K	858.8	859	859.3	
steamPlant_Sim_1sTG_3LPb.totalFeedPump.feedWaterPump	continuous	Real	K	387.4	387.4	387.4	
steamPlant_Sim_1sTG_3LPb.totalFeedPump.feedWaterSpeed_spm	continuous	Real		14.62	14.31	14.41	

Model Configuration/Case Settings

Adjustable Model Inputs

Resulting Model Outputs

Fig. A.7. ORNL SMR background presentation–Slide 7.

The Second Missing Piece... User-Tailored Interfaces

Advanced Interface - Dymola

Beginning Interface - Web Application
<http://smr-apps.ymp.nsl.gov/>

Fig. A.8. ORNL SMR background presentation–Slide 8.

The Third missing piece.... Sharing and Collaborating

The key is Web-based model development collaboration through Github with open and secure repositories for proprietary and non-proprietary work



A project to build a web-based analysis tool for small, modular nuclear reactors

72 commits 2 branches 0 releases 3 contributors

branch: master ORNL-WebSMR /

Improved layout

File	Description	Time
Documentation	Incorporating Mike's edits and a few of my own	8 months ago
Examples	minor changes to models	8 months ago
Models	Sample generated HTML	7 months ago
fmus	Pre conference material	a month ago
images	Forgot to add John's reactor image	8 months ago
schemas	Adding optional Title property	8 months ago
util	Another syntactic change to architectures.yaml	8 months ago
gignore	Improved .gignore	8 months ago
README.md	Demonstrating making changes	8 months ago
architectures.yaml	Improved layout	a month ago
fmus.yaml	More tweaking based on compiler feedback	a month ago
subsystems.yaml	More tweaking based on compiler feedback	a month ago

Code
Issues
Pull Requests
Wiki
Pulse
Graphs
Network

HTTPS clone URL
<https://github.com/ornl-smr/ornl-web-smr>

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#)

Clone in Desktop
Download ZIP

GitHub is a web-based hosting service for software development projects that use the Git revision control system. [Wikipedia]

Fig. A.9. ORNL SMR background presentation–Slide 9.

What Tools will you need?

- Modelica (open source) – for model development
- Dymola (Dassault) – for Modelica model compilation/simulation
- FMIE (Modelon) – for Excel based model simulation
- Github account – for accessing models



Fig. A.10. ORNL SMR background presentation–Slide 10.

Why Dymola?....

- Dymola provides robust solvers, libraries and interfaces that help compile modelica models efficiently

Why FMIE?....

- Models can be run in Excel without in-depth modeling expertise

Why Github?....

- Open source collaboration environment with easy access and version control.

Fig. A.11. ORNL SMR background presentation–Slide 11.

What is our initial test case?

- PRISM¹ ALMR End-to-End System model created rapidly from previous documentation models that include:
 - Reactor
 - Primary System
 - Intermediate System
 - Power Conversion
 - Grid (Generic model)
- PRISM is small sodium cooled metal fuel reactor based on the integral fast reactor design concept

¹The S-PRISM represents GEH's [Generation IV reactor](#) solution to closing the [nuclear fuel cycle](#) and is also part of its Advanced Recycling Center (ARC) proposition² to U.S. Congress to deal with [nuclear waste](#).³ It is a [sodium-cooled fast breeder reactor](#), based on the [Experimental Breeder Reactor II \(EBR-II\)](#) design, scaled up by a factor of ten. [Wikipedia]

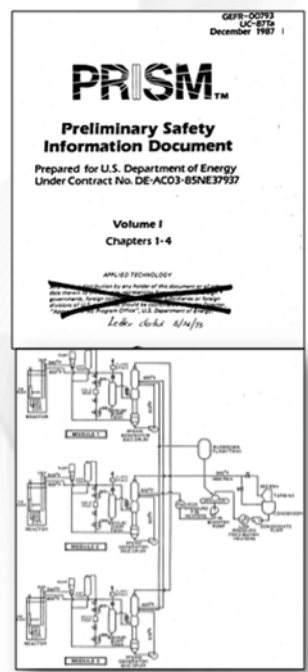


Fig. A.12. ORNL SMR background presentation–Slide 12.

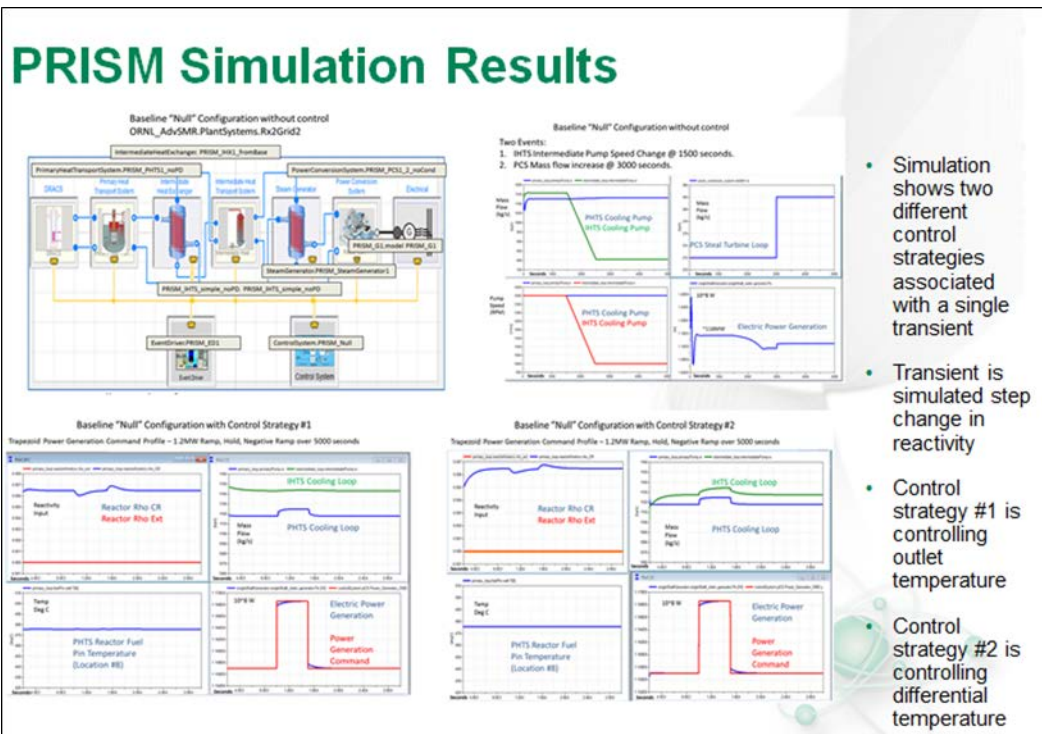


Fig. A.13. ORNL SMR background presentation–Slide 13.

Collaboration Protocols

The OpenCollaboration repository described in Section 5 is set up so all approved collaborators can read from it but NOT write to it.

When a user wants to work on the models, the user must **fork** the repository on GitHub. This will create a "forked repository" (conceptually, a copy—but one that is linked to the central repository). This repository will be owned by the individual user (not ORNL^{SMR}Modeling). This is also a kind of sandbox, but the user does not actually edit files here; the user simply publishes them here for others to see (and incorporate).

At this point, everything the collaborating user will have done is “in the cloud” (i.e., it still has nothing to do with the local computer, etc.). To actually work with the files, the user should clone the ***forked repository***. To do this, the user should use the “git clone” command to create the sandbox version on his or her computer. The user can then edit files and commit changes on his or her local computer without having any impact on the files that are in the cloud.

As a recommended practice, the user should occasionally push the changes that have been made to the user’s sandbox. This will apply those changes to the user’s ***forked repository*** and will make the changes visible to others. This is necessary if other people want to incorporate the user’s changes into what they are doing.

Based on this work flow, the following repositories exist and their purposes described.

1. ORNL^{SMR}Modeling/OpenCollaboration on GitHub – Central repository. This represents the master copy.
2. <username>/OpenCollaboration on GitHub – A repository where the users changes are stored so that OTHER PEOPLE can see them.
3. OpenCollaboration sandbox on the desktop – A place to edit files on the user’s local computer

Getting Started

To facilitate access and shorten the collaboration learning curve, a website and introductory material have been developed to aid potential partners. A description of the introductory material and the website is included as follows.

ORNL SMR Modeling Website

The project has developed an initial website that serves as a portal for information and activities associated with SMR Modelica modeling of advanced reactor concepts. The website (Fig. A.14) can be found at (<http://smrdev.ornl.gov/>). Currently, this website provides the following information and links.

- (1) A link to the ORNL Open Source GitHub repository
 - a. This repository currently contains the ALMR/FHR Modelica architecture models
- (2) A link to the Xogeny Model repository

- a. This is the original modeling repository used for the development of the web-based application and contains application files (yaml) that are necessary for the development and maintenance of the web application. This repository is currently in transition to the ORNL repository.
- (3) A link to the web-based application for modeling simulation
 - (4) Initial background material and frequently asked questions (FAQs) regarding the project
 - (5) A Quick Start Guide with links for the products needed for full collaboration (Dymola/FMI/GitHub/Git)
 - a. These include links for downloads as well as tutorials.

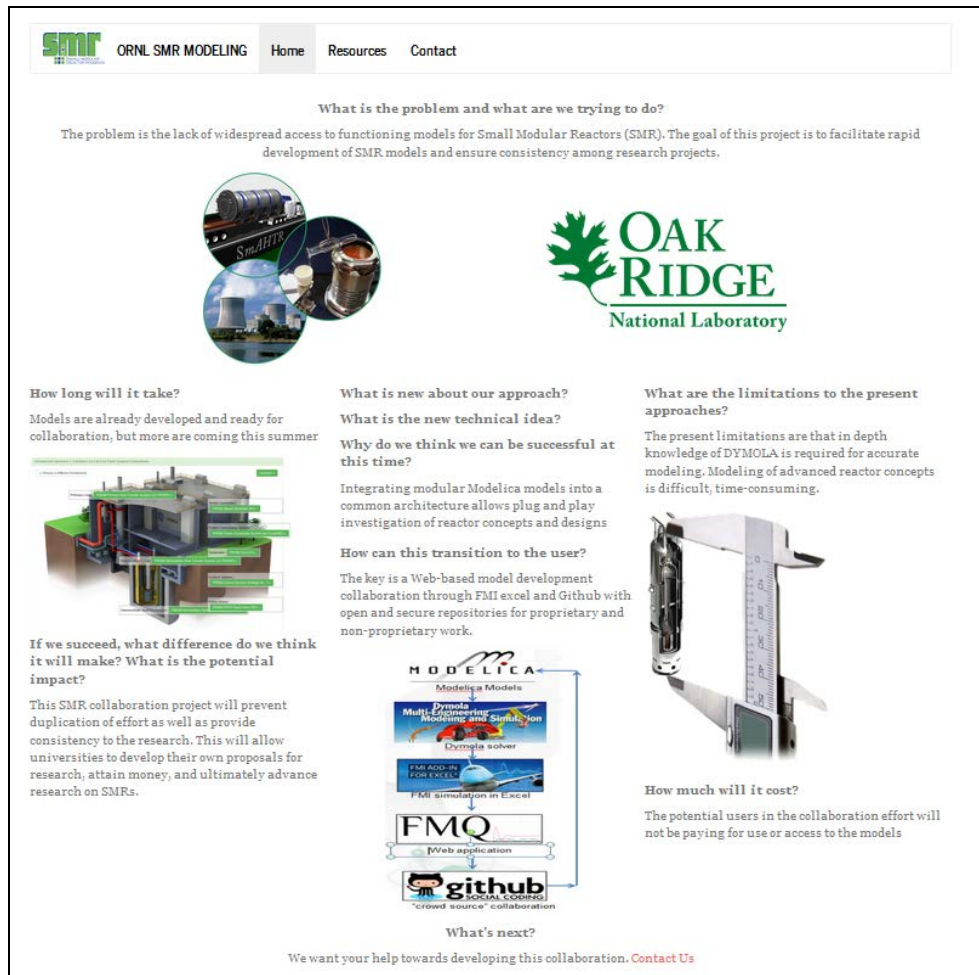


Fig. A.14. ORNL SMR modeling web site.

As a means to describe the requirements for collaboration, a Quick Start Guide has been developed. A description of the Quick Start Guide and associated useful links is presented in the section below.

Quick Start Guide

- (1) Review Brochure and Background presentation (available on the website) and send email to R. E. Hale (halere1@ornl.gov) with any questions.

- (2) Review “getting started” videos (available on the website) and send email to R. E. Hale (halere1@ornl.gov) with any questions.
- (3) Sign up for Github (<https://github.com/>)
- (4) Download and install Git (<http://code.google.com/p/tortoisegit/wiki/Download?tm=2>)
- (5) Request access to model repository (halere1@ornl.gov)
- (6) Purchase/access FMIE license from Modelon for parametric analysis (john.batteh@modelon.com)
- (7) Purchase/access Dymola license from Dessault for model development (john.batteh@modelon.com)
- (8) Create/modify models in collaboration with ORNL and submit through Github.

Each of these items will be discussed as follows.

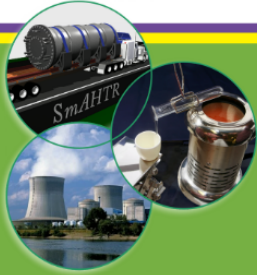
(1) Brochure

The brochure material is available on the website (<http://smrdev.ornl.gov/>) and provides an introduction to the project (Fig. A.15).

SMR Modeling and Research

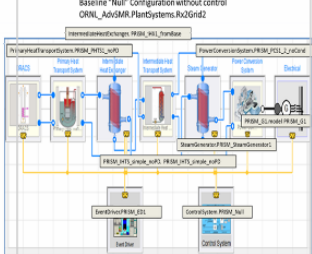
1) What is the problem and what are we trying to do?

The problem is a lack of widespread access to functioning models for SMRs. The goal of the project is to facilitate rapid development of SMR models and ensure consistency among research products.



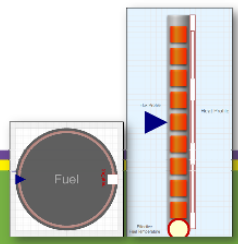
2) What are the limitations of the present approaches?

The present limitations are that in depth knowledge of DYMOLA is required for accurate modeling. Modeling of advanced reactor concepts is difficult, time-consuming,



3) What is new about our approach? What is the new technical idea? Why do we think we can be successful at this time?

Integrating modular Modelica models into a common architecture allows plug and play investigation of reactor concepts and designs.



Contact Richard Hale for SMR Collaboration
Email: halere1@ornl.gov

4) If we succeed, what difference do we think it will make? What is the potential impact?

This SMR collaboration project will prevent duplication effort as well as provide consistency to the research. This will allow universities to develop their own proposals for research, attain money, and ultimately advance research on SMRs.



PRISM is small sodium cooled metal fuel reactor

6) How can this transition to the user?

The key is Web-based model development collaboration through FMI excel and Github with open and secure repositories for proprietary and non-proprietary work.



7) How much will it cost?

The potential users in the collaboration effort will not be paying for use or access to the models.



5) How long do we think it will take?

Models are already developed and ready for collaboration, but more are coming this summer.

8) What's Next?

We want your help towards developing this collaboration. Contact Us.

Contact Richard Hale for SMR Collaboration
Email: halere1@ornl.gov

Fig. A.15. ORNL SMR modeling brochure.

(2) Background Presentation

A background presentation, along with other publically available applicable presentations and papers, is available on the website. The background presentation follows the brochure material and provides the highest level but least detailed presentation for potential collaborators to review (Figures A.1-A.13). Additional papers describing the project and application can be found under the “references” icon on the “Resources” tab on the website.

(3) Getting Started Video

Collaboration is expected to include both the development and modification of existing models, along with the use of existing models accessed through a web application that includes the use of the FMI tool explained above. To provide the necessary instructions for using the web application with the FMI tool, a video was developed that walks the user through the necessary steps. This video can be accessed through the ORNL SMR modeling web page (<http://smrdev.ornl.gov/>) under the “Resources” tab in the “Tutorials” bin. A simple tutorial on using Dymola has been prepared and is available on the website as well.

(4) Sign up for GitHub

The prospective development user needs to be registered with GitHub (<https://github.com/>). GitHub provides a common location for repositories that make use of the Git protocols. GitHub requires registration with a UserID. Following this registration, the user can request access to the ORNL SMR Modeling repository by sending an email through the ORNL SMR Modeling web page (<http://smrdev.ornl.gov/>) to the repository administrator. Upon notification of access, the user can access the models in the repository, clone the repository, and begin contributing and/or modifying models. Upon completion, the user can send a push request to the repository administrator to merge the repository models with the modified models. Criteria for acceptance of these modifications have not yet been established and will be part of the next deliverable.

(5) Download and Install TortoiseGit

Registration and access to the repository will allow a user to access models for download. This alone, however, does not allow the user to check out and modify models using Git. Full collaboration using Git allows all users to trace and track modifications with version control. This capability exists only following the installation of one of several potential versions of Git on the user’s local machine. The project choice is TortoiseGit. The instructions for installing TortoiseGit are included below. Following the installation of Git, tutorials exist within the installed Help, as well as online. A brief online tutorial can be found at the following link:

<https://help.github.com/articles/set-up-git>

Tortoise Git

The project uses Tortoise Git (Fig. A.16). The installation of Tortoise Git is described along with [a link to the download at the following website: http://code.google.com/p/tortoisegit/wiki/Download?tm=2](http://code.google.com/p/tortoisegit/wiki/Download?tm=2)

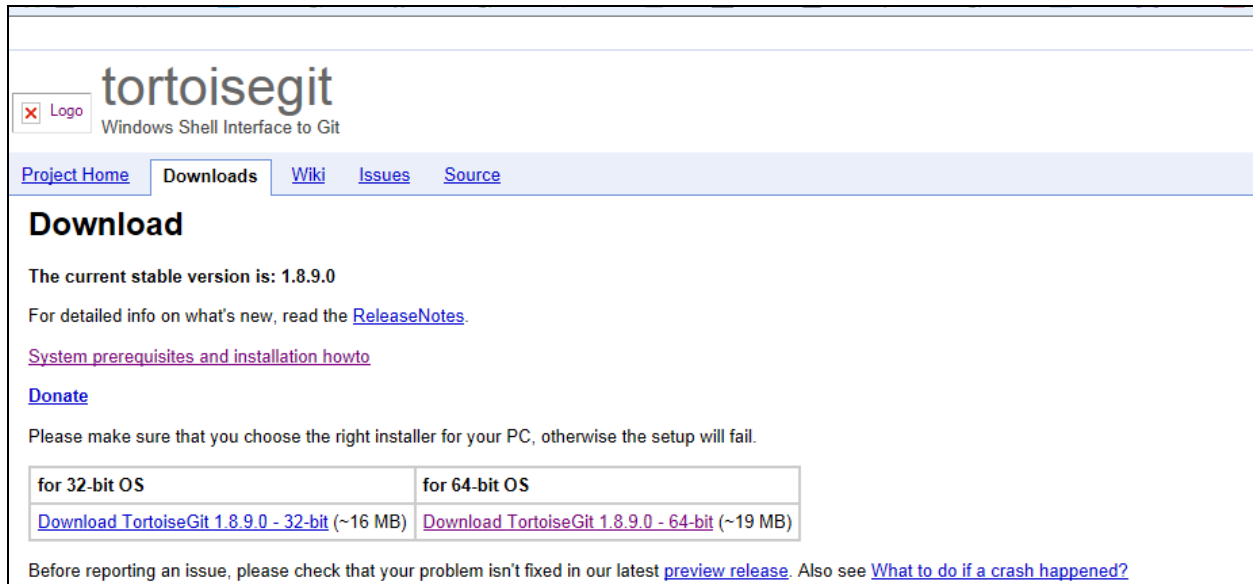


Fig. A.16. TortoiseGit download.

Simplified instructions for using Tortoise Git along with the SMR repository models are included further down in this appendix.

(6) Request access to model repository

Access to the various model repositories can be made through the ORNL SMR Modeling web page (<http://smrdev.ornl.gov/>). The “SMR Small Modular Reactor Program” icon on the “Resources” tab provides a link to the GitHub site repositories. However, until the user has obtained approval, the message displayed will indicate the user does not have access to the requested repository. Currently the only repository open to collaboration is the OpenCollaboration repository. Additional repositories will be developed as discussed in Section 5.1 and made available for collaboration as needs dictate. Access to any repository is obtained by request through the “Contact” tab on the web site.

(7) Purchase/access FMIE license from Modelon for parametric analysis

The web application requires the purchase and/or use of an FMIE license from Modelon. Currently, ORNL holds a single license. Licenses for this product are available for purchase through the Modelon web site (<http://www.modelon.com/products/fmi-add-in-for-excel/>). Additional details can be obtained and questions answered by contacting John Batteh (john.batteh@modelon.com). Currently, as a convenience to potential collaborators, negotiations are under way with Modelon for ORNL and DOE to hold several licenses that can be used by approved collaborators.

(8) Purchase/access Dymola license from Dessault for model development

Currently, the development of models requires the purchase and/or use of a Dymola license that can be purchased either directly from Dassault or through Modelon. Currently ORNL holds a single license. Licenses for this product are available for purchase through the modelon web site (<http://www.modelon.com/products/dymola/>). Additional details can be obtained and questions answered by contacting John Batteh (john.batteh@modelon.com). Currently, negotiations are planned with Dassault for ORNL and DOE to hold several licenses that can be “checked out” for use by approved collaborators.

(9) Create/modify models in collaboration with ORNL and submit through Github

Following registration on GitHub and establishing access permissions, the user is ready to check out, modify, and share a model in the repository. An example modification to a file in the ORNL SMR Modeling repository can be found further down in this Appendix. In addition to modifying models through GitHub, consideration of how best to share and package models for use within the community is needed. In the paragraphs below, some preliminary discussion of these considerations is presented.

Considerations for Using Git with Modelica/Dymola Projects

Fundamentally, Modelica files are simply text files that are interpreted by the Dymola solver. Since Git is designed for collaboration on programming projects (also a collection of text files), Git is well suited as version control software for collaborating on Modelica projects. The examples further down in the Appendix illustrate this capability. Changes to basic model parameters might involve one line of text being changed, while replacing a component entirely might be no more involved than changing six lines of the text—a testament to the powerful flexibility of the Modelica language. As demonstrated above, for changes like this, Git works flawlessly. Modelica system models however include combinations of models and libraries that are grouped into what are termed packages. Consideration of the optimum way of handling these packages is important.

Packages

Working with Git will require thought and planning as to how projects are developed within the Modelica Environment. Modelica is an object-oriented programming language, with different files constituting the working information (all the models, functions, components, etc.). The overarching “shell” is known as a package. A stylized sketch of a Modelica project might look like the following:

```
01 package widgetfactory
02 package partA
03 component widget
04 out = 1;
05 component wodget
06 out = 2;
07 function addition
08 a+b = result;
09 package partB
10 model widgetfunction
11 partA.widget A
12 partA.wodget B
13 partA.addition(A.out, B.out)
14 end widgetfactory
```

If a user is developing Modelica models, there is no fundamental reason why that user should need to use more than one high-level package, since project files can contain nested projects within them. The high-level project above is “widgetfactory” with subprojects “partA” and “partB.” In terms of the files Dymola generates when a package is saved, depending on the way in which subpackages are created, Dymola can create only a file for the highest level package, which in this case would be “widgetfactory.mo,” or it can create subpackages as folders, in which case the file structure would be widgetfactory as a folder, with two .mo files as its contents. When the highest level package is created, unchecking “Include contents of package in one file”, structure the package to be better suited for use with Git. It will be important to do this for large projects to as fine a scale as possible. Assume “widgetfactory” contains all the models developed for the modeling of advanced reactor systems—every component, function, property library,

model, test, etc. If they were all saved in one package file, and if a user wants to make a simple change for example, changing the variable “out” under widget from 2 to 3, pushing that to Git would require the entire widgetfactory to be updated. However, if partA is saved as a separate package file, then only partA.mo would change on the GitHub directory, and partB would remain unchanged. Building a coherent project architecture will be important as more users make changes to the working directories.

Revision and Merging

Another complication of working with Git and Dymola is that it is easy to make numerous unintentional but nonsubstantive changes to a package file unrelated to the substantive, intended change. For example, depending on how the package is viewed, Modelica might insert spaces in order to standardize the alignment of text. Any time a model is moved, the annotation code that describes what the GUI should display can accumulate changes that, for the most part, will not be relevant to anyone, such as the path a connector line appears to follow. There is no way to teach Git how to ignore these changes when it displays change logs. Adding to the confusion, the bigger problem that this creates is that it confuses the “diff” tool Git uses to analyze different branches of a common model. The longer two models are divergent, the more these small changes will accumulate, making branch merging potentially problematic. Ideally, the system works as follows: if user A checks out a file above and changes line 4 to out = 2 while at the same time user B has a version and decides line 4 should be out = 3, when they push the separate files to the repository, Git will make a separate branch for each changed version and will return a line-by-line comparison of the two files, looking something like

```
03 component widget
<<<<<< USER A
04 out = 2;
===== USER B
04 out = 3;
>>>>>> (log number)
```

With the number of changes that can be accumulated with simple changes, it may prove difficult to use the merge tool for significant changes, in which case, new versions might have to be uploaded to the repository as a new project file and then incorporated manually. With these considerations in mind, the establishment of a standardized method for creating models and packages will be part of the ongoing work continued during the next stage. Additional reference material for using and organizing Modelica models through the use of git can be found at the free online book at the link below:

<http://book.xogeny.com/>

Model Repositories/Branches

A single model repository is expected to be maintained by ORNL. However, multiple potential branches in the model repository are expected to be maintained based upon the pedigree and development state of the models. A listing and discussion of the purposes of these branches is included below.

Sandbox – The sandbox branch does not exist in the cloud. It is the local directory from which collaborators work. All models in this local area are in the initial stage of development with no expectation of verification or validation. No formal QA is required in this branch. This is expected to be component model focused. When these component models are ready for testing, they can be pushed to the OpenCollaboration repository and forked as a Development branch.

Development – Component models that are ready for testing reside under the Development branch. These models are tested individually and in system architectures. Test data, and documentation of expected model results, are used here to determine an initial qualification for full QA testing. Successful initial testing will result in the transfer (or forking) of these models into the QA branch.

QA – Final documented QA testing resulting in fully validated and verified models is performed in the QA branch. Successful full verification and validation (V&V) will result in models being transferred (or forked) to the Production branch.

NOTE: It is important to note that the intention of these models is for initial trade-space studies regarding operational characteristics of reactor concepts and designs. In this regard, validation and verification must be viewed in context with the intended purpose of the models. The concept of full V&V'd models within reactor licensing and safety analysis has specific and rigorous implications (i.e., Appendix B of NQA-1) which are not intended for the development of these models. V&V levels consistent with these purposes will be established in the next deliverable. For now the intention is the assurance that the models have been benchmarked against other models, or against test data without the application of rigorous NQA-1 requirements.

Production – Models that have received full V&V and are awaiting incorporation into the web application reside in the Production branch.

Collaborators will be reviewed and approved individually for access to any files. Non-QA models with no proprietary or export control concerns will be open to unrestricted access within the OpenCollaboration repository for all approved collaborators. The Development, QA, and Production branches will have access/permission restrictions determined as needed to maintain configuration control.

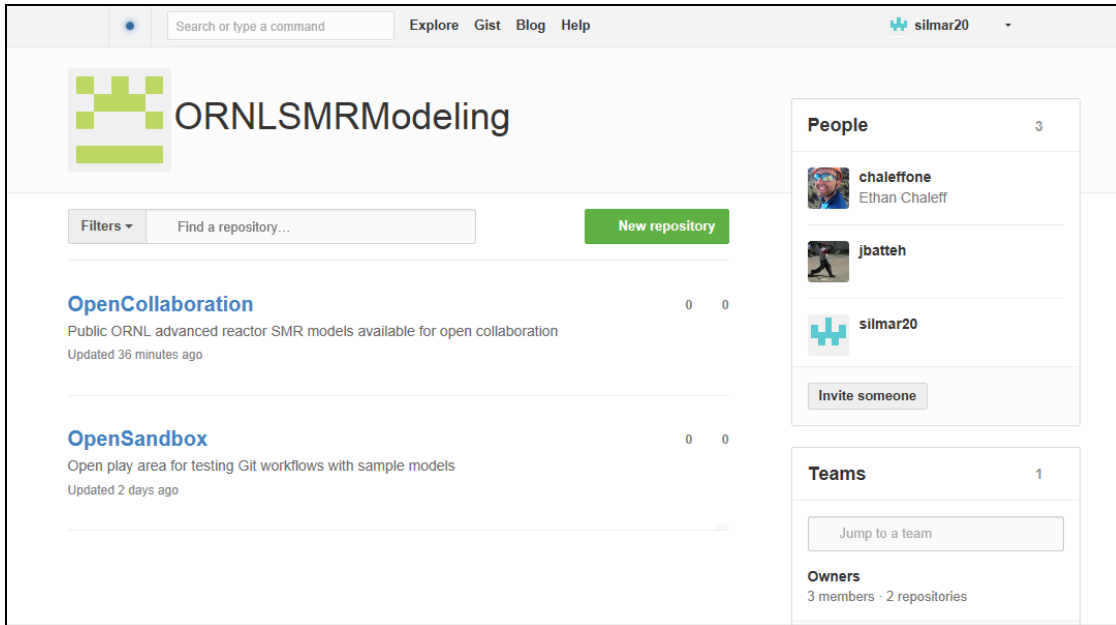


Fig. A.17. ORNL GitHub SMR modeling repository.

For the purposes of establishing the protocols for collaboration, the initial working ORNL GitHub repository is the OpenCollaboration repository (Fig. A.17). In addition to the OpenCollaboration repository, an additional OpenSandbox repository is maintained not for SMR models but as an area for testing various Git work flows.

Git Local Configuration

After (1) registering for GitHub, (2) obtaining permissions from the ORNL administrator for OpenSMRModeling GitHub access, and (3) installing TortoiseGit, the collaborating user's desktop should have the following icons shown in Fig. A.18.



Fig. A.18. Local machine Git/GitHub icons.

Once access to the ORNL GitHub site is established, double clicking the GitHub icon will provide the user a link to the online GitHub repository via the installed TortoiseGit application (Fig. A.19).

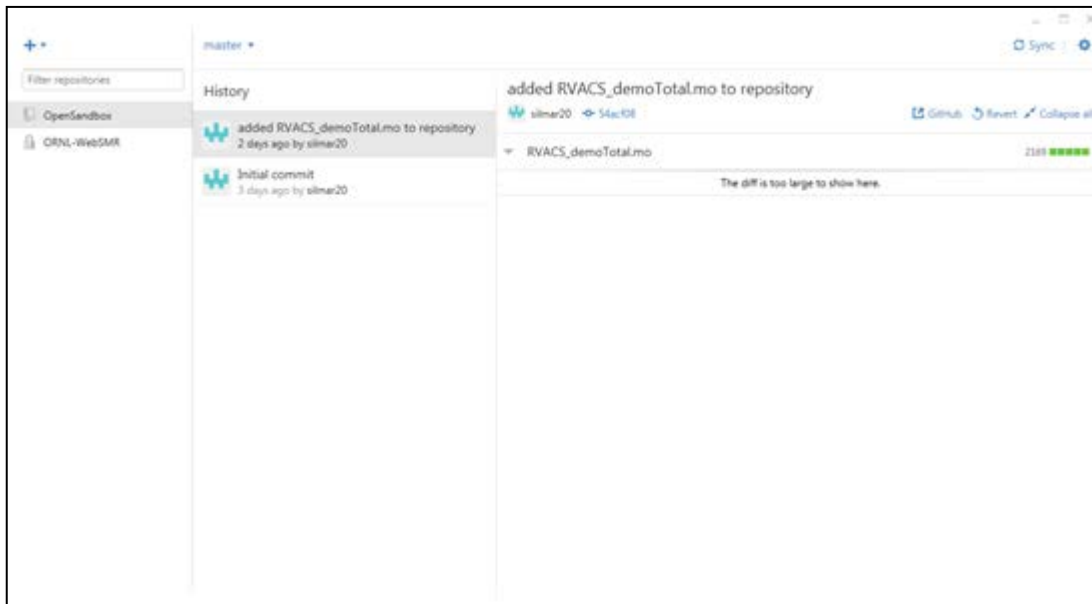


Fig. A.19. Local machine repositories (application view).

Alternatively, by clicking on the “gear” in the upper right-hand corner, the user can view the repository through a web browser (Fig. A.20).

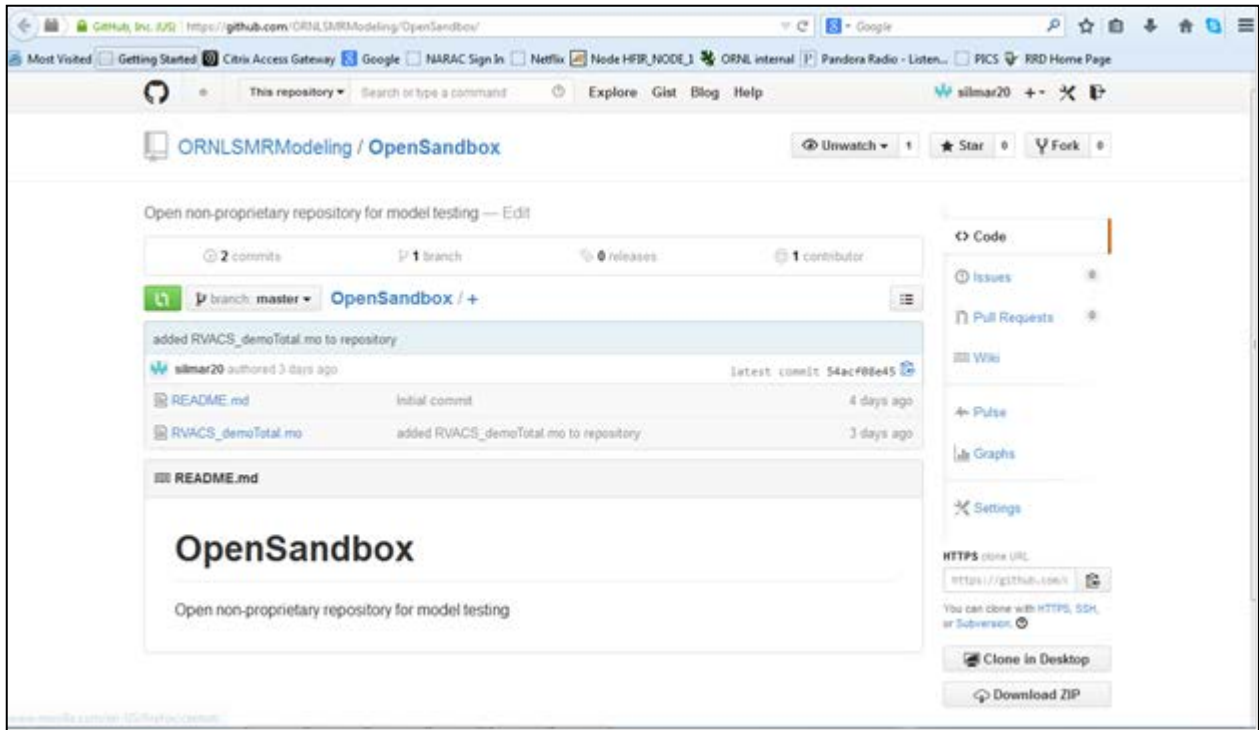


Fig. A.20. GitHub repositories (browser view).

By clicking on the “Clone in Desktop” button, the repository is “cloned” or copied to the local machine, where a local directory is maintained (Fig. A.21).

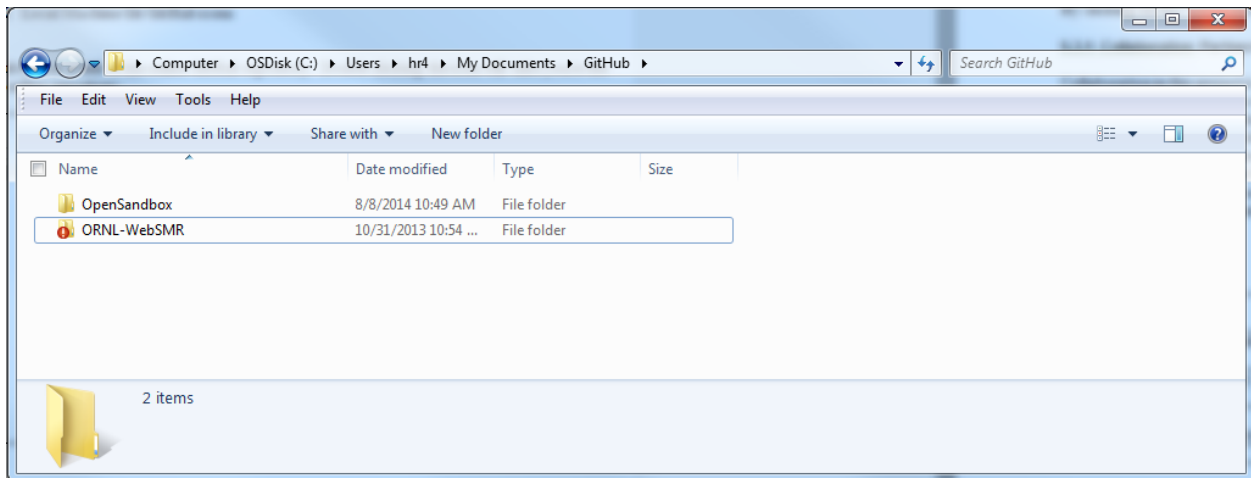


Fig. A.21. Local machine GitHub folder.

Within this folder the full range of Git commands is made available to the user through the windows application (Fig. A.22).

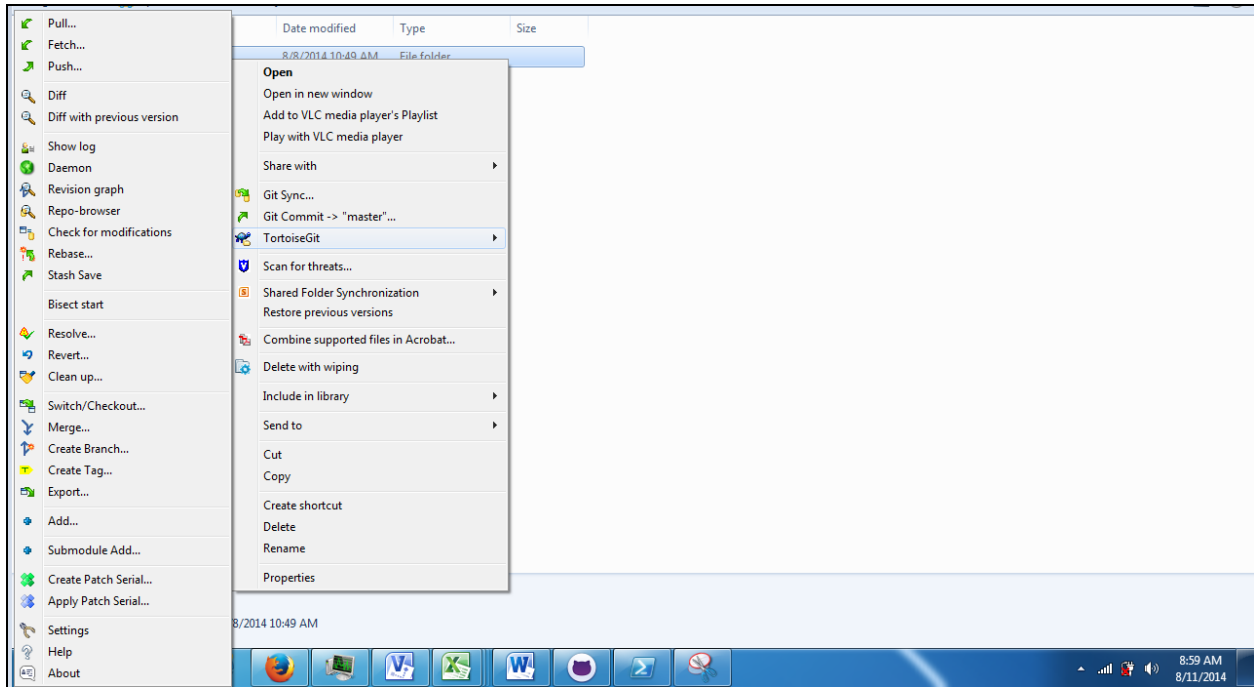


Fig. A.22. Windows Git options (right click on folder).

Alternatively, the user can perform Git commands through a command line prompt by double clicking on the GitShell icon on the desktop. The GitShell will appear (Fig. A.23).

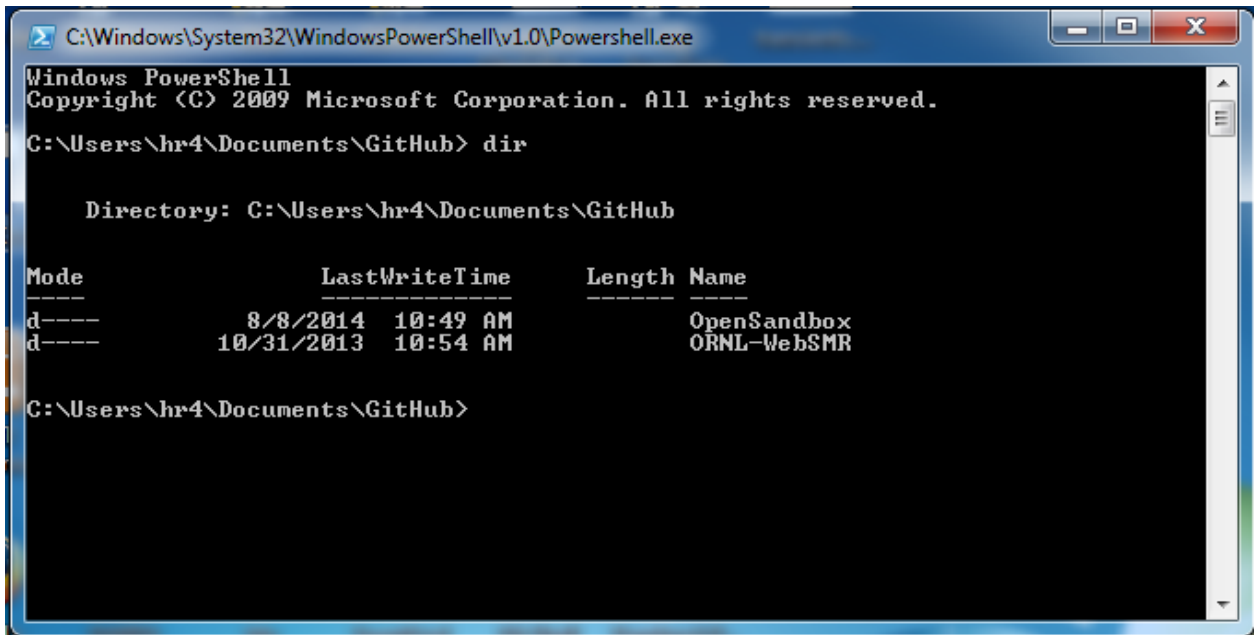


Fig. A.23. Local repositories viewed through GitShell.

These views allow the collaborator to make changes, commit, and sync with the online repository. A simple example for making these changes using these commands is presented further down in the Appendix.

Following the installation of TortoiseGit, registration with GitHub, and establishment of access to the ORNL repository, the collaborating user is ready to utilize Git to modify and share modeling changes. The changes are made and shared using the following commands (from the command line):

“DIR” :	-	directory listing of current folder
“git status” :	-	“git” status of the files in the current folder
“git add -A”	-	adds files in local directory to track changes
“git commit -m”	-	commits local changes to tracked files for pushing to repository
“git push”	-	pushes committed changes from local machine to cloud repository
“git pull”	-	pulls committed changes from cloud repository to local machine

A more in-depth discussion of these basic git commands (and others) can be found at <http://www.codeproject.com/Articles/457305/Basic-Git-Command-Line-Reference-for-Windows-Users>

A simple youtube video on the basics of using Git for software configuration control can be found at <https://www.youtube.com/watch?v=0fKg7e37bQE>

A good reference for using Git can be found at: <http://www.git-scm.com/book>

Installing Git

Along with registration on GitHub, the installation of git on the collaborator’s local machine is necessary to coordinate the modification and sharing of models in a cloud-based repository. The version of git chosen for ORNL development is TortoiseGit. This is by no means the only available version of configuration management software available for version control using the Git protocols. However, for convenience it is recommended that collaborators use TortoiseGit. TortoiseGit is available for free download at <http://code.google.com/p/tortoisegit/wiki/Download?tm=2>.

The figures below (Fig. A.24 – Fig. A.36) provide an annotated guide to the installation process.

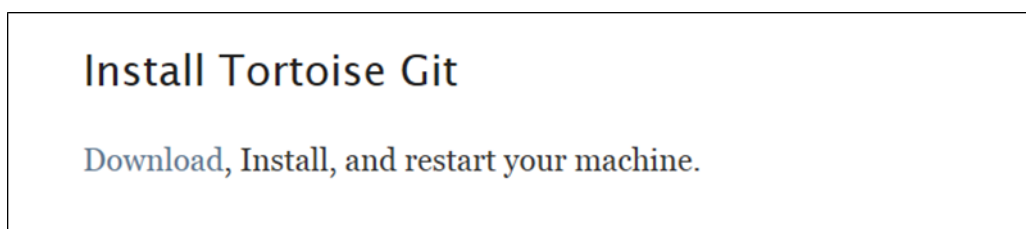


Fig. A.24. TortoiseGit download Step #1.



Fig. A.25. TortoiseGit download Step #2.

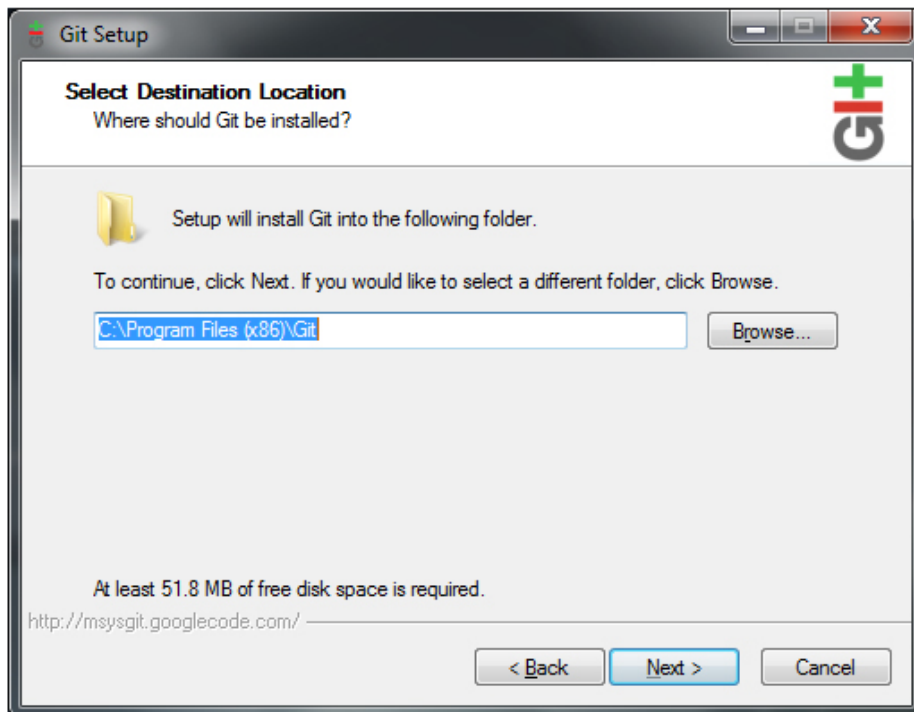


Fig. A.26. TortoiseGit download Step #3.

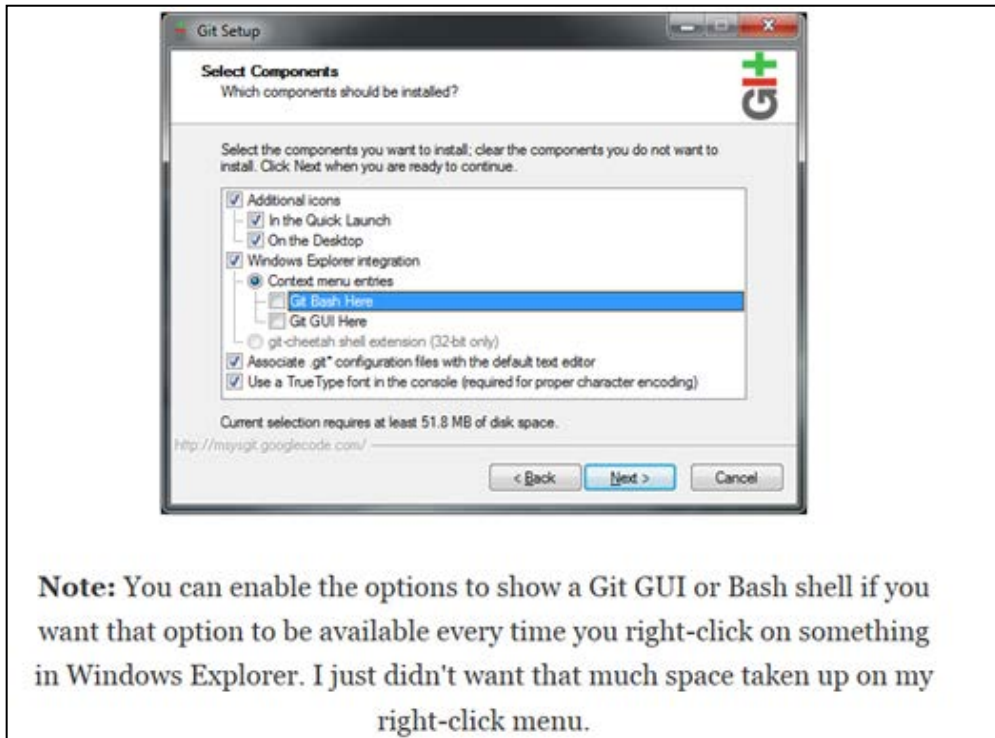
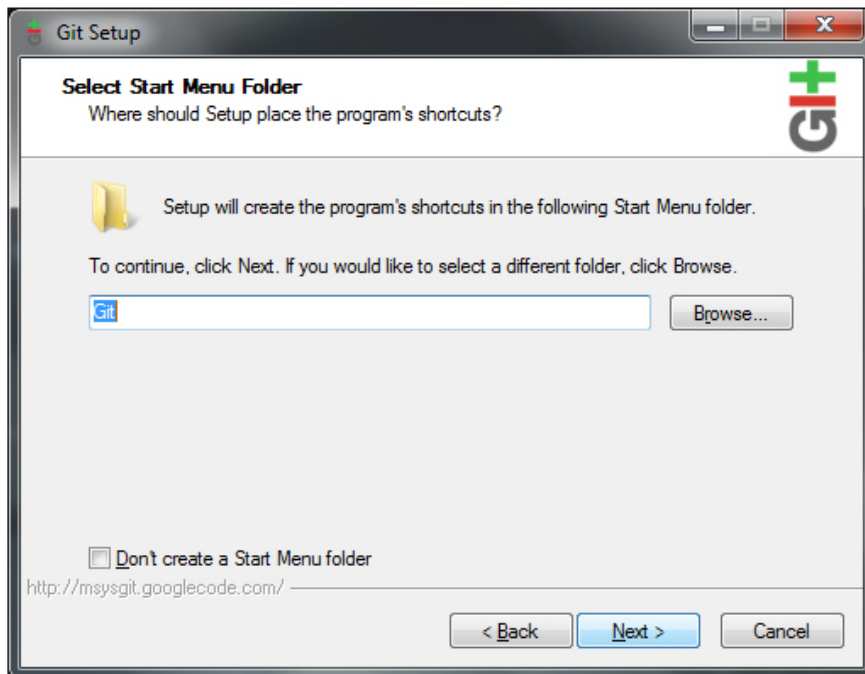


Fig. A.27. TortoiseGit download Step #4.



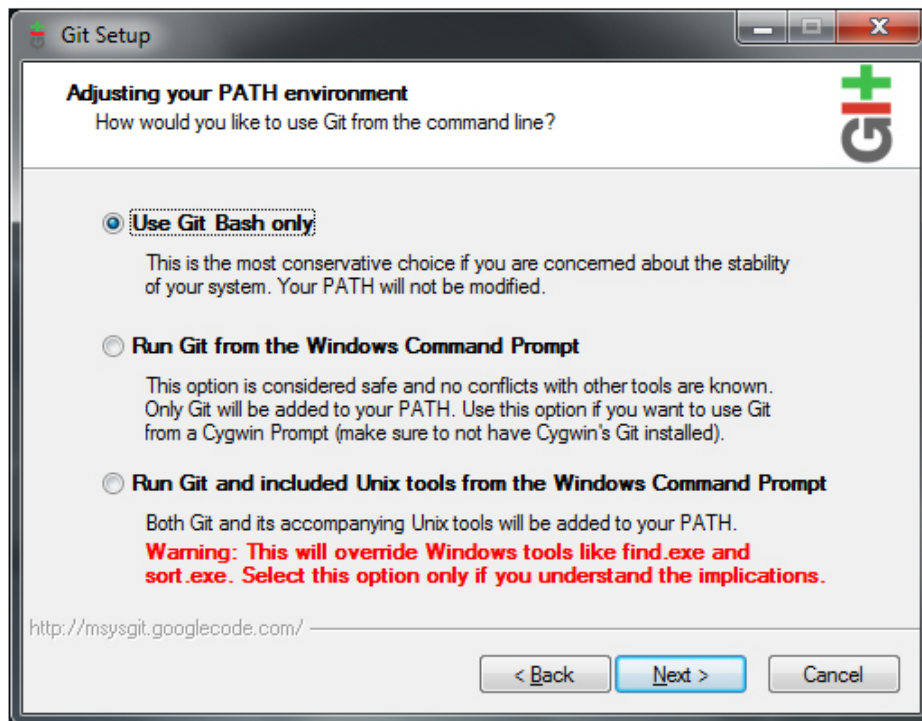


Fig. A.29. TortoiseGit download Step #6.

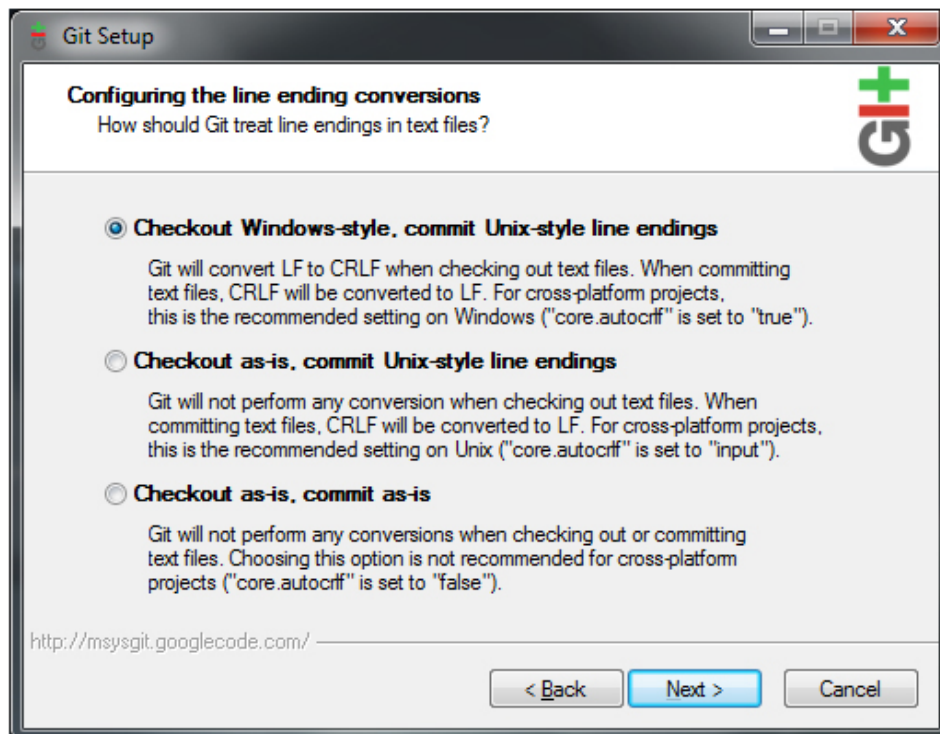


Fig. A.30. TortoiseGit download Step #7.

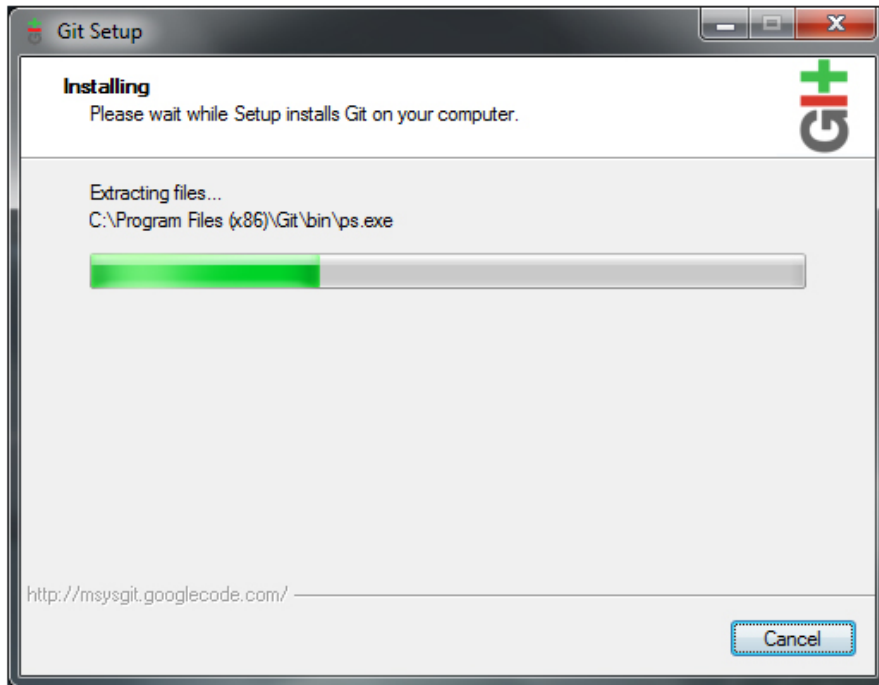


Fig. A.31. TortoiseGit download Step #8.



Fig. A.32. TortoiseGit download Step #9.

Create a new Git repository

- Create a folder -- I'm using D:\repo -- to be used for your Git repository.
- Right-click the folder and select *Git Create Repository Here*

You should now be greeted with the following message

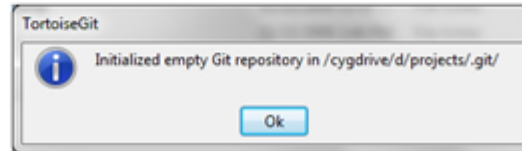


Fig. A.33. TortoiseGit tutorial–Create Repository.

Add an initial codebase to the Git repository

- Copy your source files to the git folder.
- Right click on the folder and select *Git Commit*
- In this case, I added two files. I'm going to check both of them
- click OK.

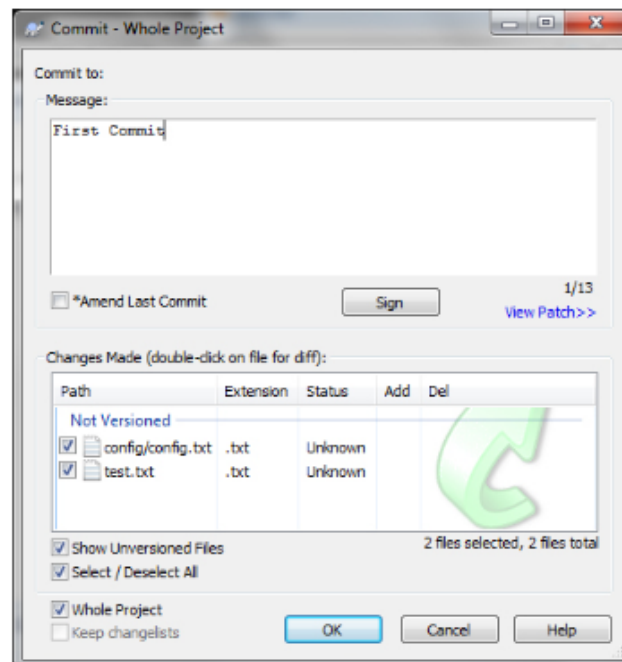


Fig. A.34. TortoiseGit tutorial–Add Codebase to Repository.

Change a file and commit

Now you can change, add, or delete files within the repository. Once you are ready to commit back to the repository, right click and select *Git Commit* just like above.

Fig. A.35. TortoiseGit tutorial–Change File and Commit.

Clone the repository

Cloning is one of the great features of Git, and while it isn't necessarily important for solo projects, it is important enough to mention here.

- Tortoise allows you to clone projects by right clicking on the desired clone folder and selecting *Clone*
- In the dialog, select the parent repository.
- Click OK
- Now, the cloned repository will be synced up with the main repository. This is useful for multi-person teams.

Fig. A.36. TortoiseGit tutorial–Clone the Repository.

(Making Model Changes with Git)

Model changes using Git are tracked and committed to allow collaboration and version control. Several steps in the process are displayed in the Figs. A.37–A.48 using the GitShell command line prompt. A good introduction to the simple use of git commands can be found at the link below.

<https://www.youtube.com/watch?v=0fKg7e37bQE>


```
C:\Windows\system32\cmd.exe

Directory of C:\Users\hr4\My Documents\GitHub
08/07/2014  02:42 PM    <DIR>          .
08/07/2014  02:42 PM    <DIR>          ..
08/08/2014  10:49 AM    <DIR>          OpenSandbox
10/31/2013  10:54 AM    <DIR>          ORNL-WebSMR
             0 File(s)      0 bytes
             4 Dir(s)  76,346,138,624 bytes free

C:\Users\hr4\My Documents\GitHub>cd OpenSandbox

C:\Users\hr4\My Documents\GitHub\OpenSandbox>git status
On branch master
Your branch is up-to-date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        RUACS_demoTotal.mo

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\hr4\My Documents\GitHub\OpenSandbox>
```

Fig. A.37. “git status” command for determining current status of files in directory.

```
C:\Windows\system32\cmd.exe

Directory of C:\Users\hr4\My Documents\GitHub
08/07/2014  02:42 PM    <DIR>          .
08/07/2014  02:42 PM    <DIR>          ..
08/08/2014  10:49 AM    <DIR>          OpenSandbox
10/31/2013  10:54 AM    <DIR>          ORNL-WebSMR
             0 File(s)      0 bytes
             4 Dir(s)  76,346,138,624 bytes free

C:\Users\hr4\My Documents\GitHub>cd OpenSandbox

C:\Users\hr4\My Documents\GitHub\OpenSandbox>git status
On branch master
Your branch is up-to-date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        RUACS_demoTotal.mo

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\hr4\My Documents\GitHub\OpenSandbox>git add RUACS_demoTotal.mo
C:\Users\hr4\My Documents\GitHub\OpenSandbox>
```

Fig. A.38. “git add” command for initiating tracking of changes in files.

```
C:\Windows\system32\cmd.exe
2 File(s)          101,738 bytes
2 Dir(s)  78,671,597,568 bytes free

C:\Users\hr4\My Documents\GitHub\OpenSandbox>git commit -m "added RVACS_demoTotal.mo to repository"
[master 54acf08] added RVACS_demoTotal.mo to repository
 1 file changed, 2169 insertions(+)
 create mode 100644 RVACS_demoTotal.mo

C:\Users\hr4\My Documents\GitHub\OpenSandbox>git push
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

When push.default is set to 'matching', git will push local branches
to the remote branches that already exist with the same name.

In Git 2.0, Git will default to the more conservative 'simple'
behavior, which only pushes the current branch to the corresponding
remote branch that 'git pull' uses to update the current branch.

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Username for 'https://github.com': silmar20
Password for 'https://silmar20@github.com':
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 23.14 KiB | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/ORNL/SMRModeling/OpenSandbox.git
 156e9b5..54acf08  master -> master

C:\Users\hr4\My Documents\GitHub\OpenSandbox>
```

Fig. A.39. “git commit -m” command for committing changes in the file to the local repository and providing explanation (i.e., -m “added RVACS_demoTotal.mo to repository”).

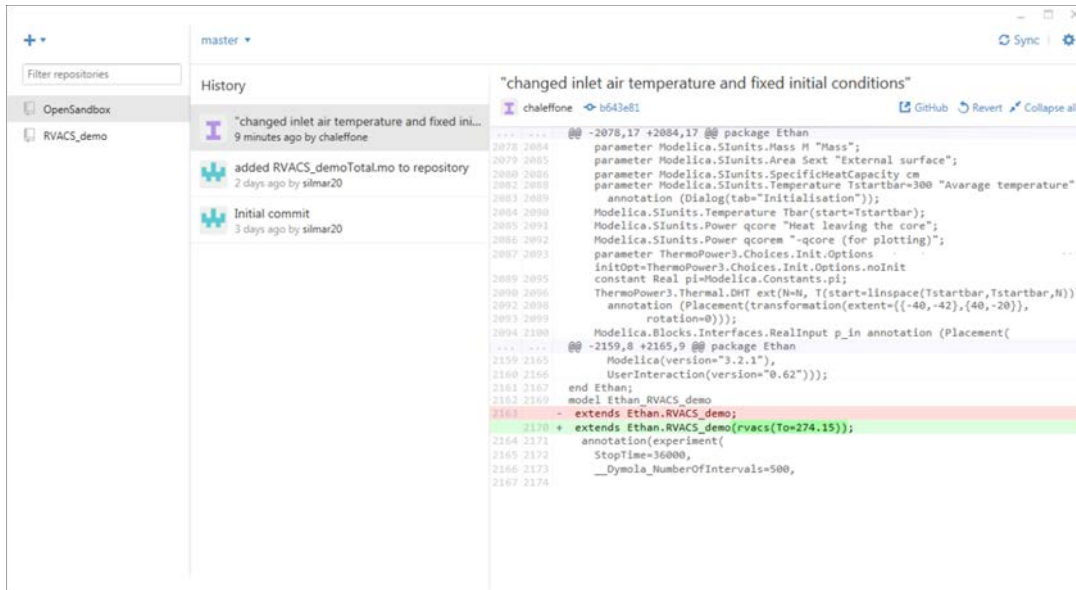


Fig. A.40. Local history and view of changes to file.

Figure A.40 shows a simple change of a parameter a user might make to see the effect of inlet air on the RVACS model. The change performed is as follows: By default, an instance of a Modelica class is instantiated with its default values (the line “extends Ethan.RVACS_demo”), which in this case happens to be an air temperature of 298.15 K. Changing the default value requires a modification of the class instantiation, which is done by telling Modelica to load the class Ethan.RVACS_demo, but changing the value of “To” in the component “rvacs” from whatever its default is to the specific value of 274.15.

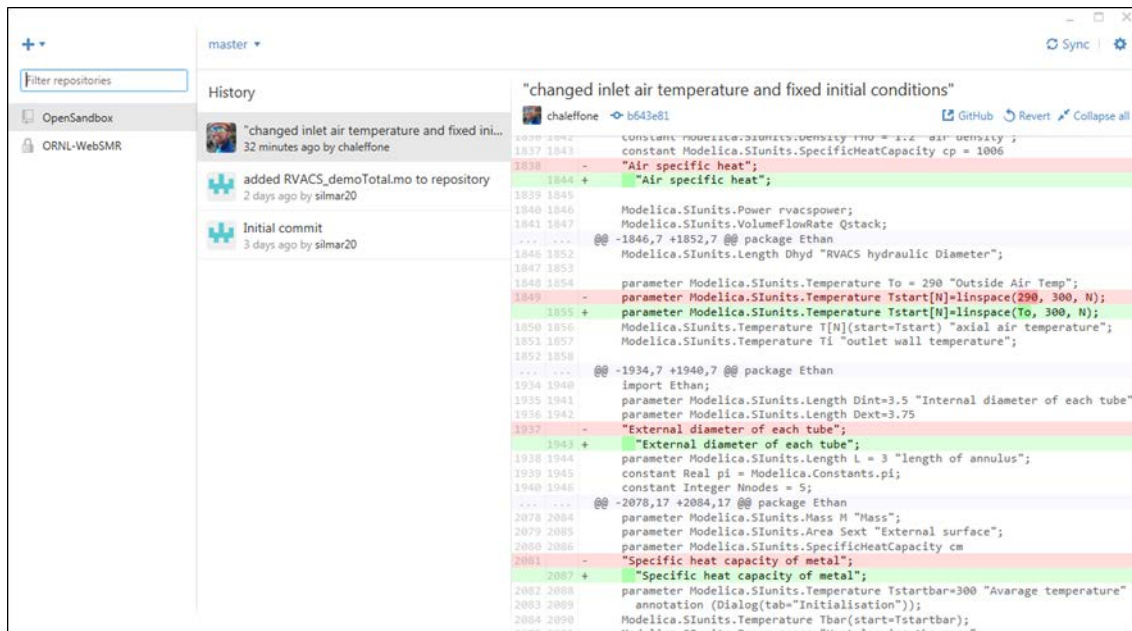


Fig. A.41. Local history and view of changes to file.

```

ThermoPower3.Thermal.MetalTube vessel(
  rext=Dint/2,
  L=L,
  rhomcm=1000000,
  rint=Dint/2 - 0.025,
  N=Nnodes,
  lambda=25,
  Tstartbar=573.15)  a;

```

Fig. A.42. Modelica model change in steel density.

After the changes have been committed and pushed to GitHub (Fig. A.43), they constitute the current working version of the model. Typing `git status` into the command line lets you know that there is one change on the repository that is not reflected in your current working directory. The command `git pull` (Fig. A.44) updated the current git directory.

```

posh-git ~ OpenSandbox [master]
Directory: C:\Users\hr4\Documents\GitHub\OpenSandbox

Mode                LastWriteTime         Length Name
----                -
-a---              8/7/2014   2:42 PM             79 README.md
-a---              8/11/2014  11:04 AM          102030 RUACS_demoTotal.mo

C:\Users\hr4\Documents\GitHub\OpenSandbox [master +0 ~1 -0] > git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   RUACS_demoTotal.mo
#
no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\hr4\Documents\GitHub\OpenSandbox [master +0 ~1 -0] > git add -A
C:\Users\hr4\Documents\GitHub\OpenSandbox [master +0 ~1 -0] > git commit -m
"changed density of steel"
[master 925d2f5] changed density of steel
1 file changed, 10 insertions(+), 6 deletions(-)
C:\Users\hr4\Documents\GitHub\OpenSandbox [master] > git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 498 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To https://github.com/ORNL/SMRModeling/OpenSandbox.git
 b643e81..925d2f5  master -> master
C:\Users\hr4\Documents\GitHub\OpenSandbox [master] >

```

Fig. A.43. Git “commit” of change to model.

```
posh-git ~ OpenSandbox [master]
C:\Users\eao\Documents\GitHub> ls

Directory: C:\Users\eao\Documents\GitHub

Mode                LastWriteTime         Length Name
----                -
d-----            8/11/2014  10:19 AM             OpenSandbox
d-----            7/30/2014   3:04 PM             RVACS_demo

C:\Users\eao\Documents\GitHub> cd OpenSandbox
C:\Users\eao\Documents\GitHub\OpenSandbox [master]> ls

Directory: C:\Users\eao\Documents\GitHub\OpenSandbox

Mode                LastWriteTime         Length Name
----                -
-a----            8/11/2014   9:42 AM             79 README.md
-a----            8/11/2014  10:19 AM          101817 RVACS_demoTotal.mo

C:\Users\eao\Documents\GitHub\OpenSandbox [master]> git status
# On branch master
# Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
#   (use "git pull" to update your local branch)
#
nothing to commit, working directory clean
C:\Users\eao\Documents\GitHub\OpenSandbox [master]> git pull
Updating b643e81..925d2f5
Fast-forward
 RVACS_demoTotal.mo | 16 ++++++++-----
 1 file changed, 10 insertions(+), 6 deletions(-)
C:\Users\eao\Documents\GitHub\OpenSandbox [master]>
```

Fig. A.44. Git “pull” of change to model to update repository.

Although issuing commands in the command line interface is easy and fast, the use of the GitHub GUI allows better visualization of the changes made. In the UI for GitHub we can see that “rhomcm” has been changed from 4e6 to 1e6 (Fig. A.45).

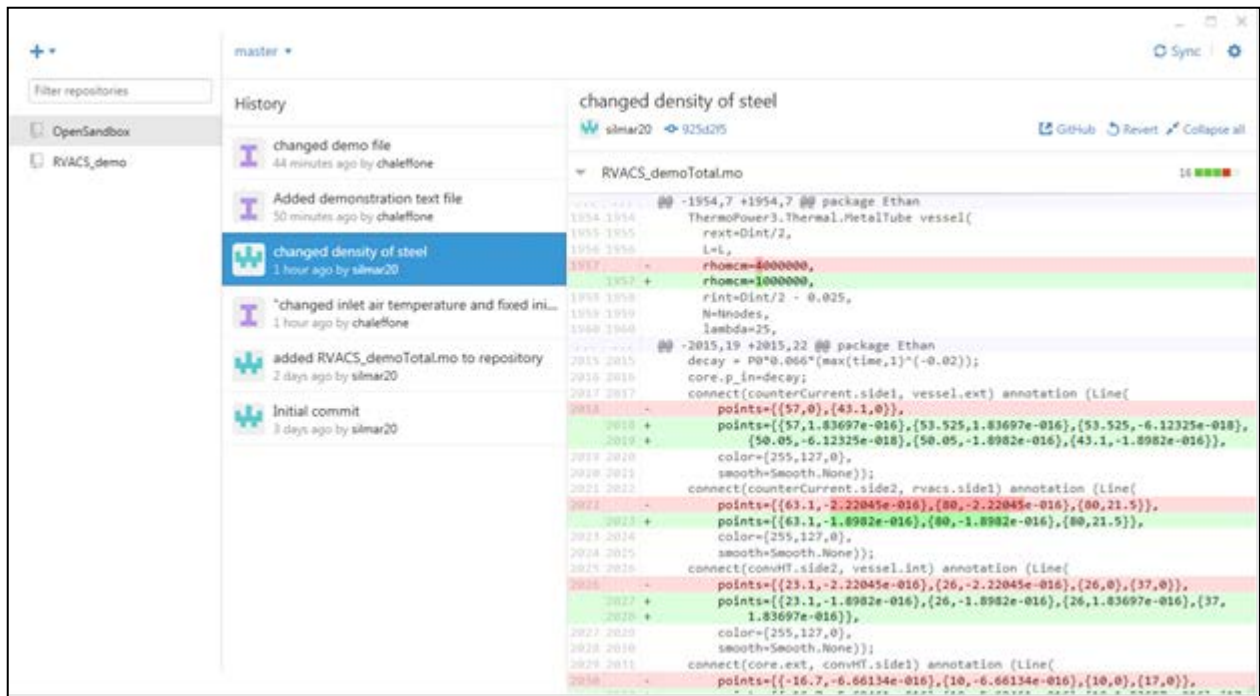


Fig. A.45. TortoiseGit GUI denoting density model changes.

By reloading the package in Modelica (accomplished easily with File>Recent Files) you can see that the change has been reflected in the model (Figs. A.46–A.48).

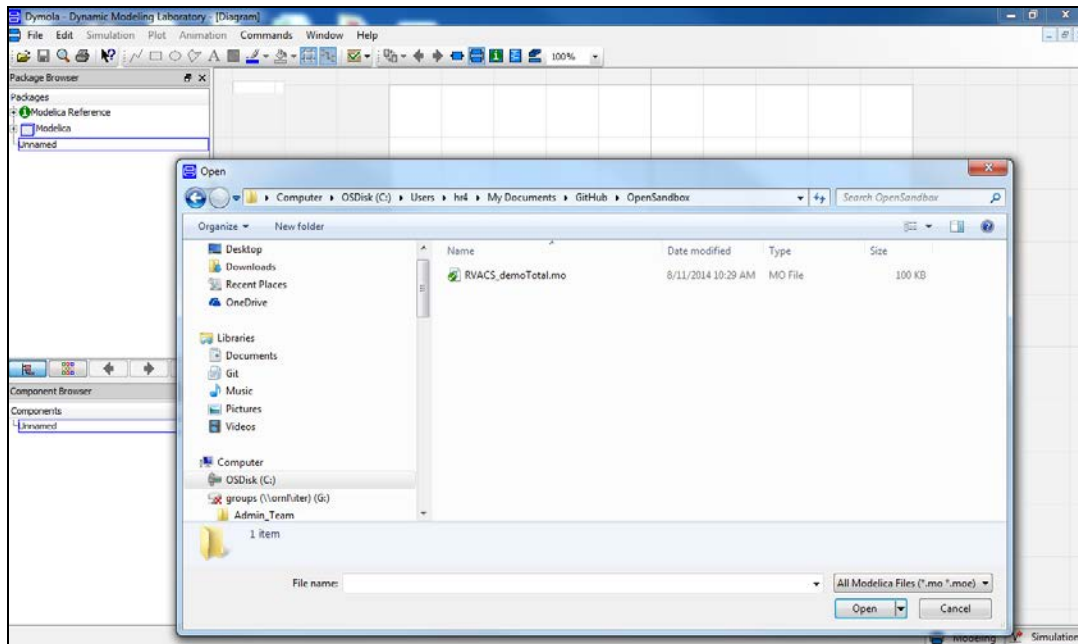


Fig. A.46. Dymola model modified (date modified).

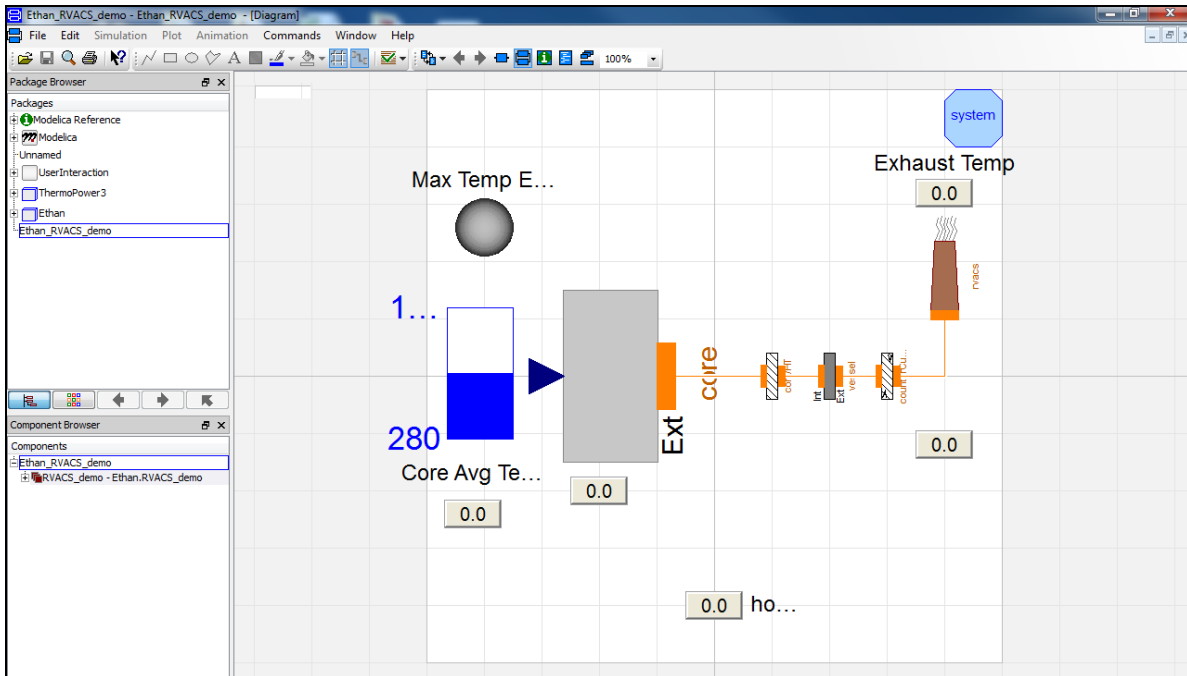


Fig. A.47. Dymola model.

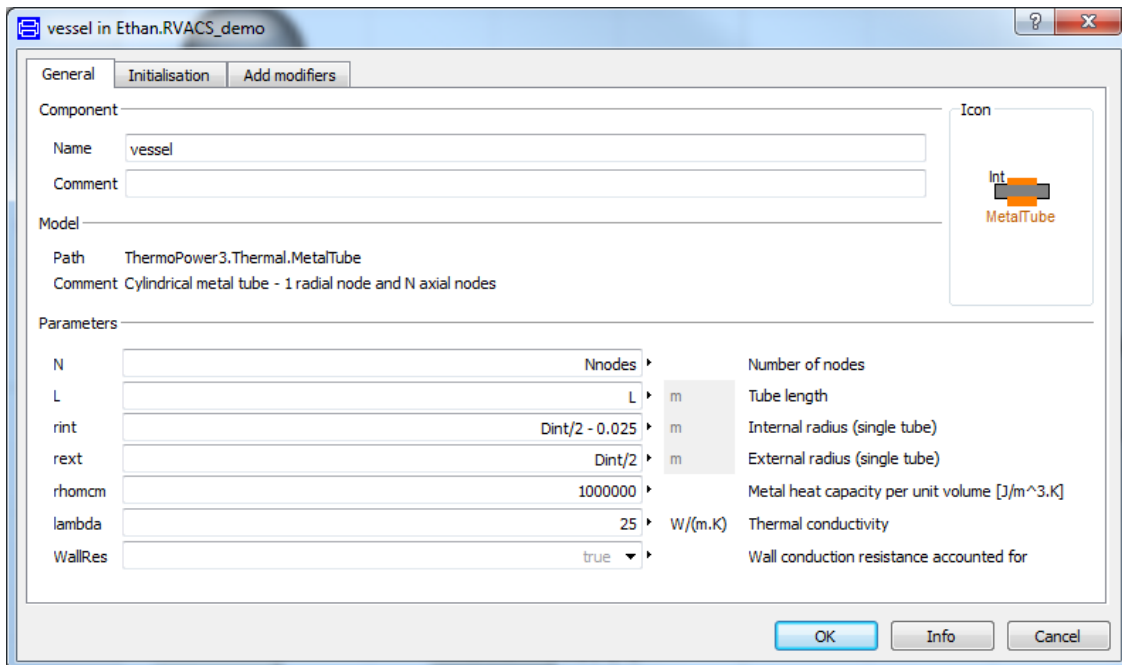


Fig. A.48. Model modification reflected in Dymola parameter interface.