ORNL/TM-2019/1314

NEAMS Workbench Status and Capabilities

Robert A. Lefebvre
Brandon R. Langley
Paul Miller
Marco Delchini
Mark L. Baird
Jordan P. Lefebvre

September 30, 2019

Approved for public release.
Distribution is unlimited.

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

ORNL/TM-2019/1314

Reactor Nuclear Systems Division

NEAMS WORKBENCH STATUS AND CAPABILITIES

Robert A. Lefebvre
Brandon R. Langley
Paul Miller
Marco Delchini
Mark L. Baird
Jordan P. Lefebvre

Date Published: September 2019

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

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

2D, 3D	two-dimensional, three-dimensional
ANL	Argonne National Laboratory
ARC	Argonne Reactor Computation (suite)
CASL	Consortium for Advanced Simulation of Light Water Reactors
CFD	computational fluid dynamics
DDI	Definition-Driven Interpreter
DOE	US Department of Energy
FPL	fuels product line
GUI	graphical user interface
HALITE	hierarchical input template expansion engine
HIVE	hierarchical input validation engine
INL	Idaho National Laboratory
IPL	integration product line
LANL	Los Alamos National Laboratory
LWR	light-water reactor
MCNP	Monte Carlo N-Particle
MOOSE	multiphysics object-oriented simulation environment
NCSU	North Carolina State University
NEAMS	Nuclear Energy Advanced Modeling and Simulation (DOE program)
ORNL	Oak Ridge National Laboratory
PBS	portable batch system
RPI	Rensselaer Polytechnic Institute
RPL	reactors product line
SAM	system analysis module
SHARP	simulation-based high-efficiency advanced reactor prototyping (tools)
SNL	Sandia National Laboratories
UNF-ST&DARDS	Used Nuclear Fuel – Storage, Transportation & Disposal Analysis Resource and Data System
VERA	Virtual Environment for Reactor Applications
VII	VERA Input Interpreter
WASP	workbench analysis sequence processor

ABSTRACT

The mission of the US Department of Energy's Nuclear Energy Advanced Modeling and Simulation (NEAMS) program is to develop, apply, deploy, and support state-of-the-art predictive modeling and simulation tools for the design and analysis of current and future nuclear energy systems. This is accomplished by using computing architectures that range from laptops to leadership-class facilities. The NEAMS Workbench is a new initiative that will facilitate the transition from conventional tools to high-fidelity tools by providing a common user interface for model creation, review, execution, output review, and visualization for integrated codes. The Workbench can use common user input, including engineering-scale specifications that are expanded into application-specific input requirements through the use of customizable templates. The templating process can enable multifidelity analysis of a system from a common set of input data. Additionally, the common user input processor can provide an enhanced alternative application input that provides additional conveniences over the native input, especially for legacy codes. Expansion of the integrated codes and application templates available in the Workbench will broaden the NEAMS user community and will facilitate system analysis and design. Current and planned capabilities of the NEAMS Workbench are detailed here.

1. INTRODUCTION

The mission of the US Department of Energy's (DOE's) Nuclear Energy Advanced Modeling and Simulation (NEAMS) program is to develop, apply, deploy, and support state-of-the-art predictive modeling and simulation tools for the design and analysis of current and future nuclear energy systems. This is accomplished by using computing architectures ranging from laptops to leadership-class facilities. The tools in the NEAMS ToolKit will enable transformative scientific discovery and insights otherwise not attainable or affordable and will accelerate the solutions to existing problems, as well as the deployment of new designs for current and advanced reactors. These tools will be applied to solve problems identified as significant by industry, and consequently, they will expand the validation, application, and long-term utility of these advanced tools.

The NEAMS program is organized into three product lines: Fuels Product Line (FPL), Reactors Product Line (RPL), and Integration Product Line (IPL).

NEAMS FPL and RPL provide advanced tools, such as the (1) BISON and MARMOT fuel performance tools based on the Multiphysics Object-Oriented Simulation Environment (MOOSE) from Idaho National Laboratory [1,2], and (2) the PROTEUS neutronics code and the NEK5000 computational fluid dynamics code from the Simulation-based High-efficiency Advanced Reactor Prototyping (SHARP) tools from Argonne National Laboratory (ANL) [3]. However, these advanced tools often require large computational resources, can be difficult to install, and require expert knowledge to operate, causing many analysts to continue to use traditional tools instead of exploring high-fidelity simulations.

The NEAMS IPL is responding to the needs of the design and analysis communities by integrating robust multiphysics capabilities and current production tools in an easy-to-use common analysis environment. This enables end users to apply high-fidelity simulations to inform lower order models used for advanced nuclear system design, analysis, and reactor licensing.

The NEAMS Workbench is an initiative that will facilitate the transition from conventional tools to high-fidelity tools by providing a common user interface for model creation, review, execution, output review, and visualization for integrated codes [4]. The Workbench can provide a common user input, including engineering-scale specifications that are expanded into code-specific input requirements through the use of customizable templates. The templating process can enable multifidelity analysis of a system from a common set of input data. Expansion of the codes and application templates available in the Workbench will broaden the NEAMS user community and will facilitate system analysis and design. Users of the Workbench will still need to license and install the appropriate computational tools, but the Workbench will provide a more consistent user experience and will ease the transition from one tool to the next. This report incorporates details from previous years [5] and includes progress made in the 2019 fiscal year.

2. MULTIFIDELITY PHYSICS

The NEAMS Workbench will enable analysts to choose from a variety of tools integrated from many projects and code teams. A conceptual design of tools can be integrated for advanced reactor analysis, as shown in Figure 1. Current production tools with advanced components supported by the NEAMS program such as those from the Argonne Reactor Computation suite (ARC) [6] and the SCALE Code System [7] are shown in light gray; Monte Carlo N-Particle (MCNP)® [8] and other production tools are shown in dark gray; tools from the NEAMS ToolKit are shown in maroon; and tools from the Consortium for Advanced Simulation of Light Water Reactors (CASL) Virtual Environment for Reactor Applications (VERA) [9] are shown in black. The NEAMS Workbench will make it possible to use the same fundamental engineering input data to create code-specific input models for each tool. This helps enable analysts to learn more about specific phenomena by performing reference high-fidelity analyses. This will confirm approximations or assumptions in fast-running lower order design calculations.

Multiphysics capabilities provided by tools such as SHARP and MOOSE will continue to provide fully coupled capabilities under the Workbench. The Workbench also provides for traditional single physics codes that form a multiphysics suite with coupling through manipulation of the output of one code to serve as input for the next. Translation tools will be developed and integrated into a workflow manager to extract and format the needed data in a manner that does not require intrusive changes to the physics codes themselves.

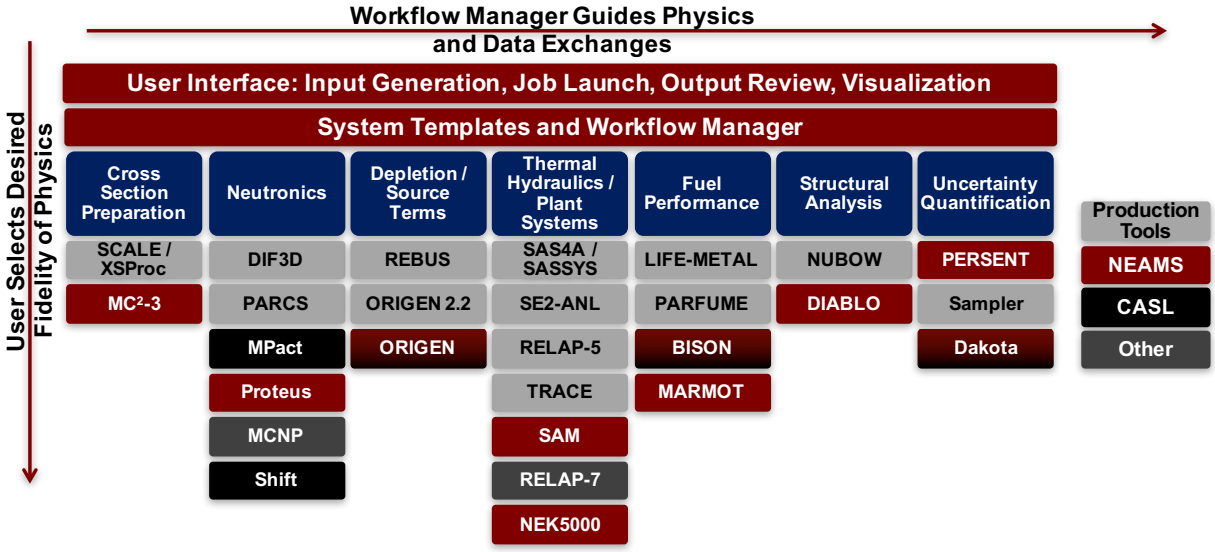

Figure 1. Conceptual design of tools integrated in the NEAMS Workbench for advanced reactor analysis.

3. USER INTERFACE

To enable users to easily transition from one tool to another, an intuitive user interface is desired. Such an interface will guide new or infrequent users on the proper use of a tool without impeding the experienced users in rapidly generating or modifying inputs. Visualization of input geometry and data files is desired, and the user interface should also provide easy access to computed results in text, tabular, and graphical forms. All integrated codes should be presented with a similar look and feel, and the ability to compare results from multiple analyses should be available.

The rapid development and deployment of the NEAMS Workbench was enabled by leveraging the Fulcrum user interface, which was developed over several years and was included in the 2016 release of SCALE 6.2. Fulcrum introduced several new concepts as an integrated user interface for nuclear analysis. It builds on decades of experience from thousands of users, and it replaces eight user interfaces from the 2011 release of SCALE 6.1. Fulcrum is a cross-platform graphical user interface (GUI) designed to create, edit, validate, and visualize input, output, and data files. Fulcrum directly connects the user with the text form of the input file while providing inline features to assist with building the correct inputs. Fulcrum provides input editing and navigation, interactive geometry visualization, job execution, overlay of mesh results within a geometry view, and plotting of data from file formats. An error checker interactively identifies input errors such as data entry omissions or duplications for all supported codes. The input validation engine identifies allowed data ranges and interdependencies in the input and then reports inconsistencies to the user. The layout of panels in Fulcrum is highly configurable to accommodate the preferences of users, as shown in Figure 2.

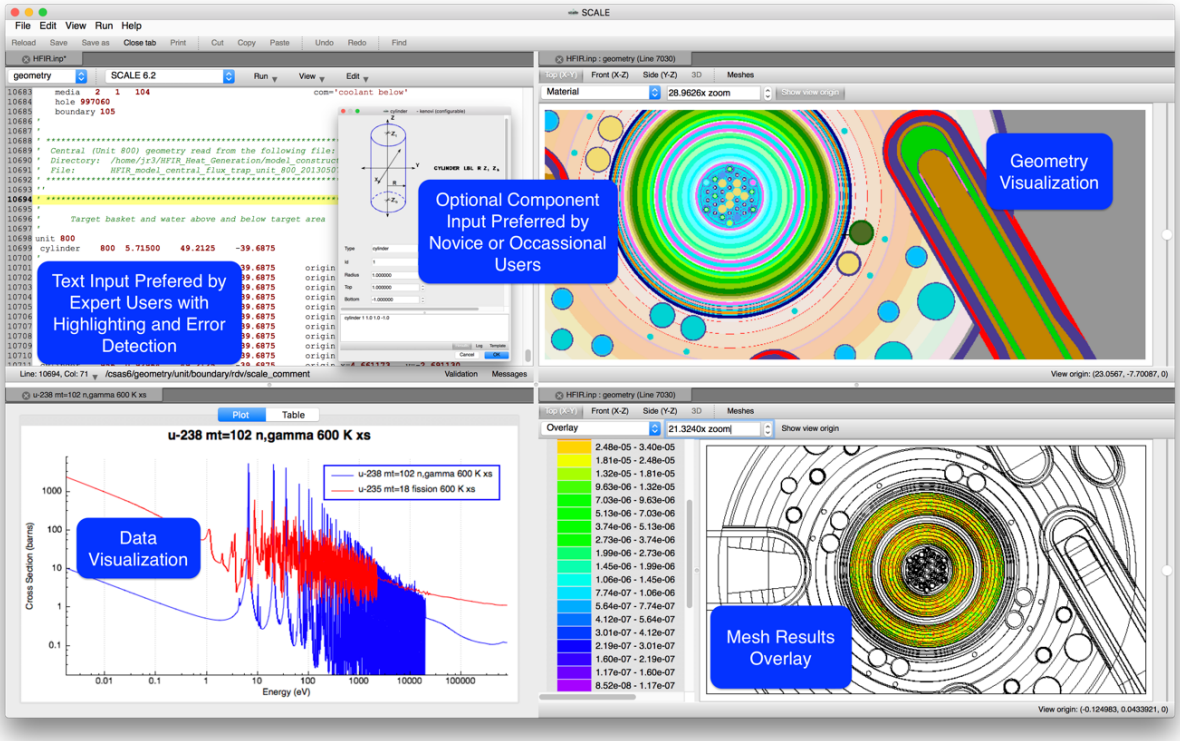

Figure 2. The NEAMS Workbench leverages the Fulcrum user interface from SCALE.

The configuration files that define how each code input is structured, the requirements and interdependencies of each input block, and the types and content of the dialog boxes available for input assistance are all provided in text files that can be created and edited by any Workbench user. In this way, the members of any team can integrate their tools into the Workbench, whether they are widely available production tools, custom-developed analysis tools in a proprietary environment, or university-led prototypes for the exploration of new algorithms. Custom-developed configuration files can be contributed back to Workbench developers to be considered for inclusion in the production version. An example of the interactive input error detection is shown for a SCALE input in Figure 3, where a user inadvertently typed `u02` (u-zero-2) instead of `uo2` to specify uranium dioxide as a material. The first error at the bottom of the screen demonstrates how Fulcrum compares the current input against the list of allowed values in this context and recommends alternatives to assist the user. A second error is shown where the user was specifying the uranium enrichment for this material. Here an input rule is applied stipulating that the weight percentages must sum to 100%. In this case, the user inadvertently entered a total of 101%, and Fulcrum identified that error, as well.

Leveraging the configuration files for each code supported by the Workbench, the Fulcrum user interface provides context-aware assistance in auto-completing the input, which can be useful when learning the intricacies of a new code or when transitioning between many codes, as is the intent of the Workbench. An example from SCALE is shown in Figure 4. In the left image, the user inserts the cursor at the desired location in the input, presses the key combination of *ctrl-space*, and Fulcrum lists all available options in this context, providing geometry shapes in this example. Selecting a *configurable* option launches an additional interactive dialog, shown on the right, to provide additional assistance in populating this portion of the input.

The Fulcrum interface currently supports visualization capabilities from SCALE for geometry, with results overlaid as shown in Figure 5. Plotting capabilities are available for nuclear data and covariance data plotting, as well as a variety of computed quantities using line plots, histograms, bar charts, and surface plots, as shown in Figure 6. The NEAMS Workbench is a natural evolution of the Fulcrum user interface. The visualization capabilities of Fulcrum have been expanded beyond those required by SCALE, especially to support analysis with NEAMS finite element codes, by integrating visualization capabilities from VisIt [10]. The integration of the ParaView [11] visualization tool is in testing.

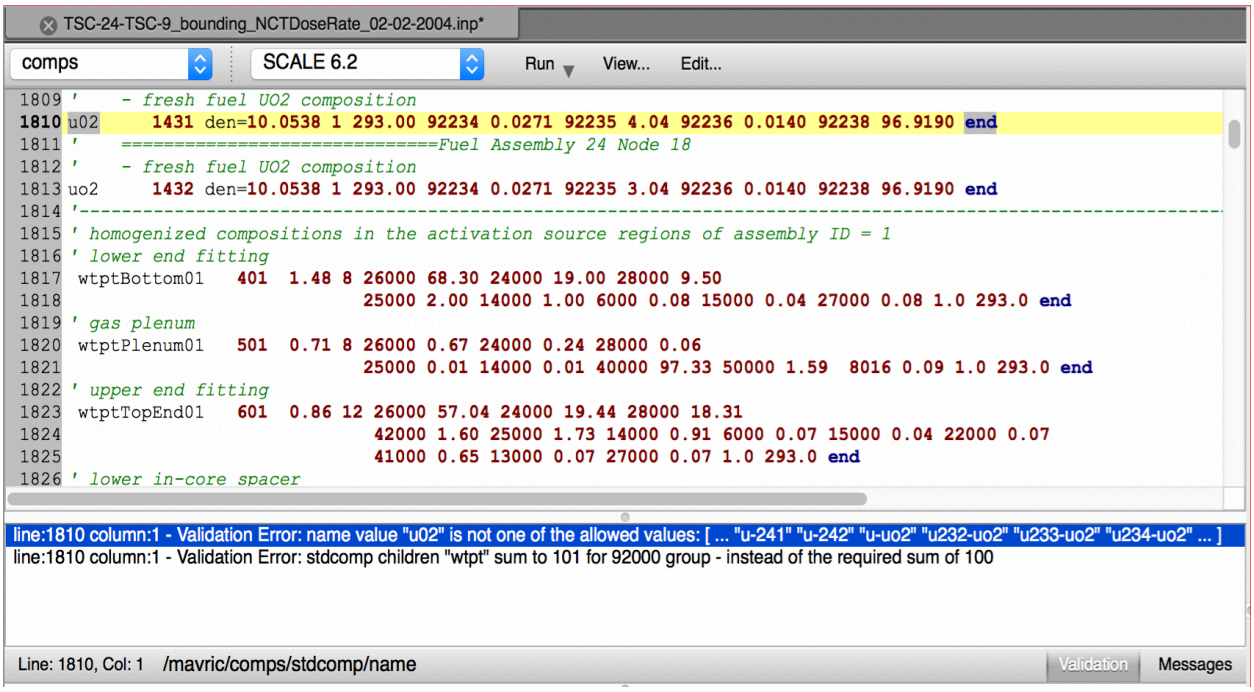

Figure 3. Fulcrum user interface with interactive input error detection.

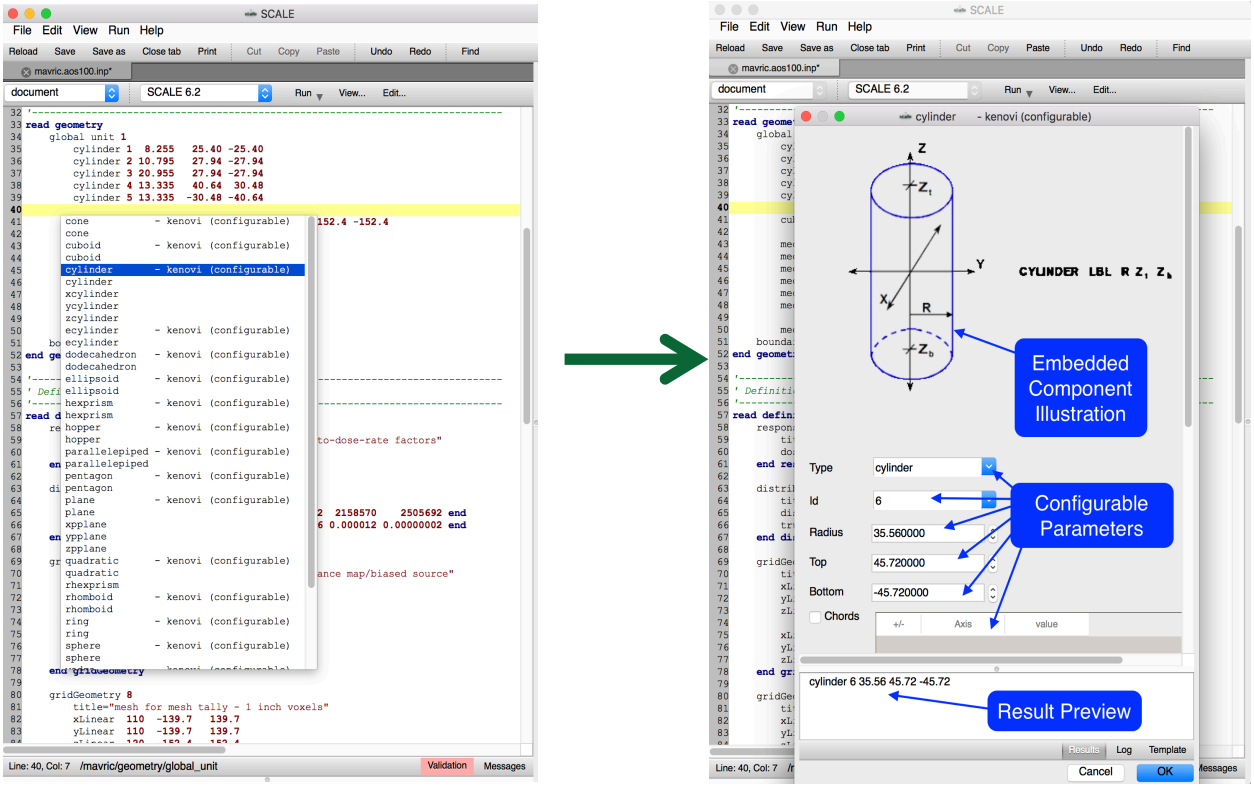

Figure 4. Fulcrum auto-completion and forms-based input.

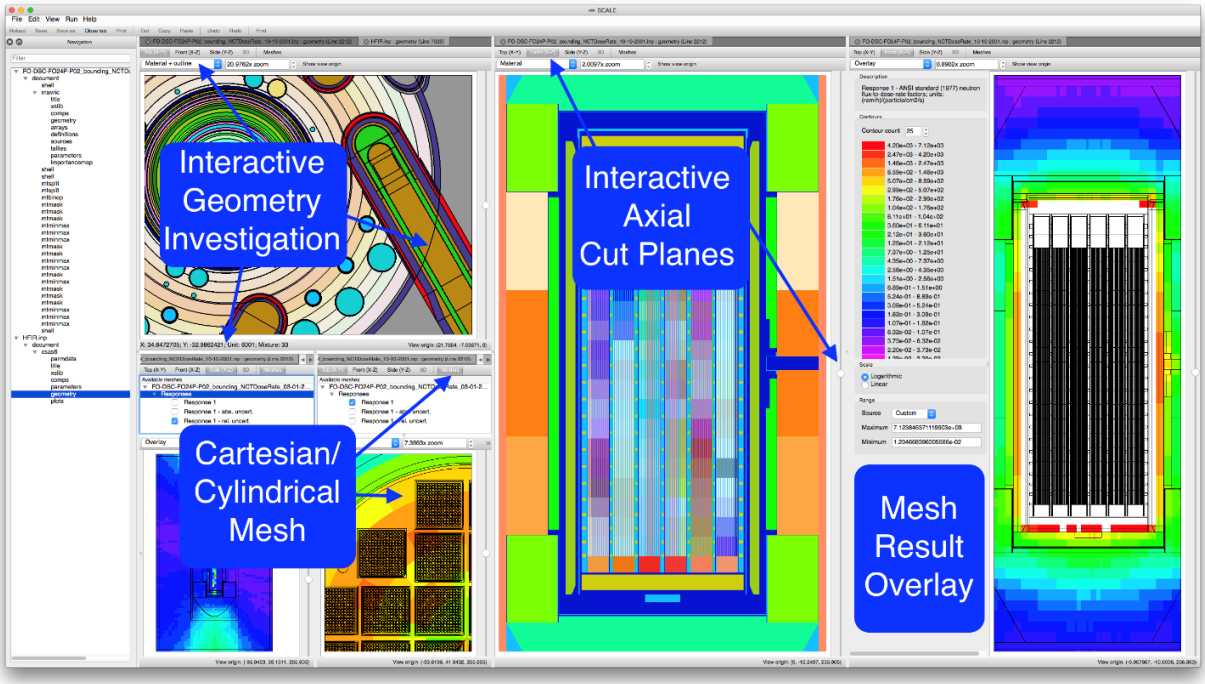

Figure 5. Fulcrum geometry visualization capabilities.

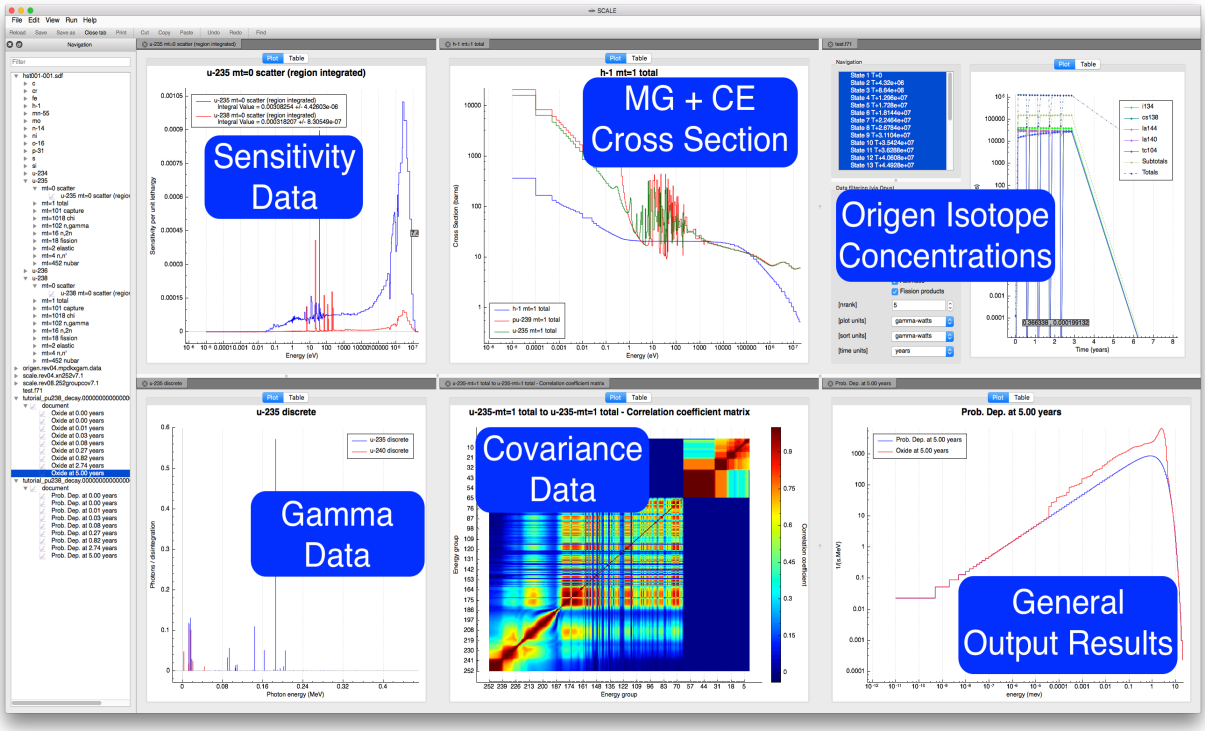

Figure 6. Fulcrum plotting capabilities.

4. INPUT CONFIRMATION AND ANALYSIS TEMPLATES

The use of multiple tools within the NEAMS Workbench is facilitated through the automatic confirmation of the input format or syntax, input-schema-allowed values, and completeness, as well as through templates that provide for a common definition of a system that can be expanded and run with multiple tools. The combination of the input syntax, schema, templates, and syntax highlighting is called the application grammar. The grammar is a fundamental integration point between the application and the NEAMS Workbench.

The Workbench Analysis Sequence Processor (WASP) [12] package is an open source C++ library and toolset for streamlining the lexical and semantic analysis of ASCII-formatted inputs. WASP includes lexers, parsers, and interpreters to provide input processing of integrated applications. WASP uses a parse-tree data structure to facilitate input data retrieval and validation. The Language Server Protocol [13], an open standard for integrated development environments, is new in 2019. WASP provides the Workbench with the ability to process common input and data formats. WASP also provides input analysis and templating capabilities through the Hierarchical Input Validation Engine (HIVE) and the Hierarchical Input Template Expansion Engine (HALITE).

4.1 HIERARCHICAL INPUT VALIDATION ENGINE

With HIVE, a combination of 19 rules is used to describe valid input and to interactively provide users with feedback on the validity of their input. The rules are implemented in an input schema for each supported code. Given an input schema and an input parse-tree from WASP, HIVE will conduct simple type, value, and occurrence checks, as well as complex relational checks. As integrated in the Workbench, HIVE provides immediate feedback to the user in the *Validation* panel, as previously shown in Figure 3. The HIVE schema files can be stored inside the Workbench application in text format, so they can be supplemented or customized by expert users if desired. Additionally, if the integrated application or the application's runtime wrapper can generate the HIVE schema, the Workbench will cache these files for future examination and use. The runtime layer is discussed in Section 6.

The 19 rules of HIVE are:

1. MinOccurs: describes the minimum number of times that an element is allowed to appear under its parent context;

2. MaxOccurs: describes the maximum number of times that an element is allowed to appear under its parent context;

3. ValType: describes the allowed value type for the element (Int, Real, String);

4. ValEnums: describes a list of allowed value choices for the element;

5. MinValInc: describes the minimum inclusive value that this element is allowed to have if it is a number (the provided input value must be greater than or equal to this);

6. MaxValInc: describes the maximum inclusive value that this element is allowed to have if it is a number (the provided input value must be less than or equal to this);

7. MinValExc: describes the minimum exclusive value of the element in the input if it is a number (the provided input value must be strictly greater than this);

8. MaxValExc: describes the maximum exclusive value of the element in the input if it is a number (the provided input value must be strictly less than this);

9. ExistsIn: describes a set of lookup paths into relative sections of the input file and possible constant values where the value of the element being validated must exist;

10. NotExistsIn: describes a set of lookup paths into relative sections of the input file where the value of the element being validated must not exist;

11. SumOver: describes what sum the values must add to under a given context;

12. SumOverGroup: describes what sum the values must add to under a given context when grouped by dividing another input element's value by a given value;

13. IncreaseOver: describes that the values under the element must be increasing in the order that they are read;

14. DecreaseOver: describes that the values under the element must be decreasing in the order that they are read;

15. ChildAtMostOne: describes one or more lists of lookup paths into relative sections of the input file (and possible values) where at most one is allowed to exist;

16. ChildExactlyOne: describes one or more lists of lookup paths into relative sections of the input file (and possible values) where at exactly one is allowed to exist;

17. ChildAtLeastOne: describes one or more lists of lookup paths into relative sections of the input file (and possible values) where at least one must exist;

18. ChildCountEqual: describes one or more lists of lookup paths into relative sections of the input file where the number of values must be equal; and

19. ChildUniqueness: describes one or more lists of lookup paths into relative sections of the input file where the values at all these paths must be unique.

These rules facilitate all input validation tasks except for higher order validation tasks such as comparing input data with data from a related binary file or validating constructive solid geometry parameters. At this time, the validation engine cannot handle recursion (e.g., validation of mathematical expression). Additional training material, with examples, is provided in the NEAMS Workbench at `Help>Documents>SESSION2-InputValidation.pdf` drop-down menu.

4.2 HIERARCHICAL INPUT TEMPLATE EXPANSION ENGINE

HALITE couples flat or hierarchical data with application-specific input templates to facilitate Workbench input auto-completion and future workflow tasks. The HALITE engine provides Workbench with the capability to create application-specific input using application-agnostic data. HALITE provides common attribute and expression substitution, as well as the unique capability to drive input expansion via a single hierarchical data set. This is the same data-driven workflow construct that has been successfully demonstrated in the Used Nuclear Fuel – Storage, Transportation & Disposal Analysis Resource and Data System (UNF-ST&DARDS) [14].

HALITE has the following templating features:

- Scalar and iterative attribute and mathematical expression substitution enables traditional and complex data insertion;
- Substitution formatting enables rigid fixed-width input formats, where needed;
- Conditional template blocks enable inclusion or exclusion of template sections using some condition;
- File input enables reuse of a common sub-template; and
- Parameterized file input enables advanced reuse of a sub-template with explicit loop or implicit array element iterative repetition or specific data sets.

A sketch of the HALITE workflow to expand a common problem definition in terms of engineering parameters such as dimensions, compositions, etc., into code-specific inputs is show in Figure 7.

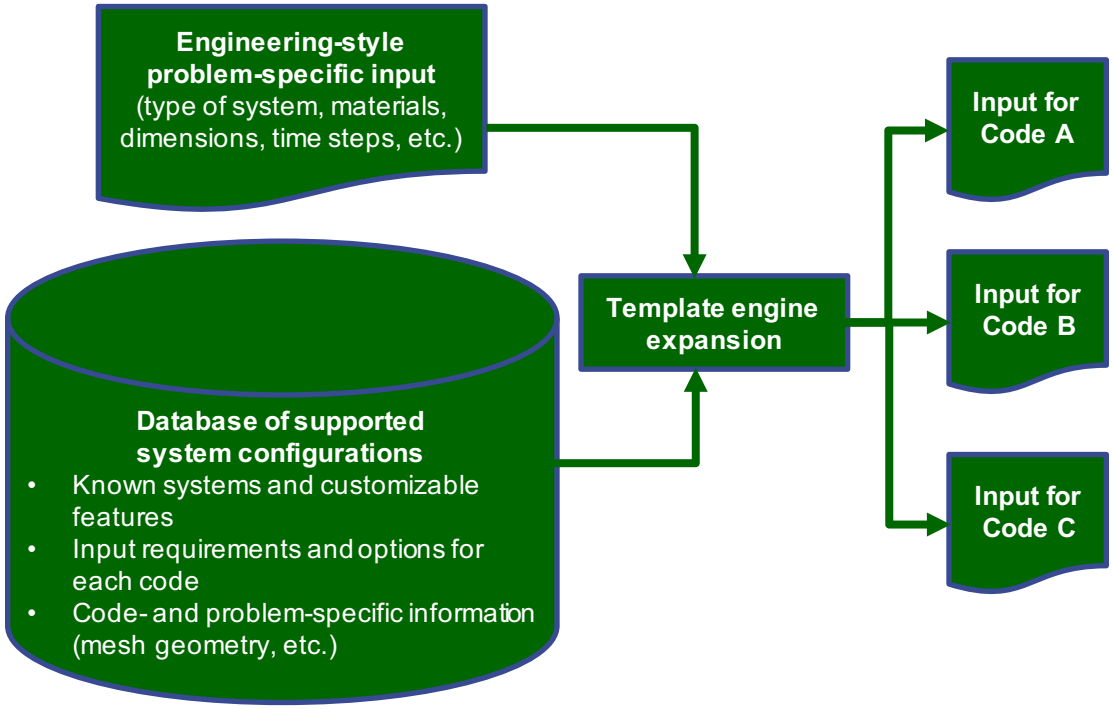

Figure 7. Sketch of HALITE template expansion to provide input for multiple codes from the same problem definition.

5. VISUALIZATION TOOLS

The interpretation of input data, geometry models, and output results is enabled through the convenient integration of multiple visualization tools. The VisIt tool is embedded in the Workbench for visualization of 2D and 3D mesh geometry and results. Customized 2D plotting is enabled for any application using the native plotting package inherited from Fulcrum that has been extended with a customizable file processor interface that enables plotting data embedded in text file input or output. In addition to VisIt and extendable processor result visualization, Workbench also supports visualization of several notable data formats inherited from Fulcrum, including the covariance and ORIGEN data, as depicted in Figure 8.

Figure 8. General 2D data visualization in the NEAMS Workbench.

5.1 VisIt VISUALIZATION TOOLKIT

VisIt is a powerful distributed, parallel visualization and graphical analysis tool developed for analysis of mesh data from high-performance computing. In the NEAMS Workbench, VisIt is composed of three parts:

1. **The VisIt GUI**, which will load as an embedded application in its own dockable panel. The native VisIt controls are accessible from the VisIt GUI.

2. **The VisIt canvas**, which is a tab within the NEAMS Workbench workspace layout.

3. **VisIt visualization**, which requires a VisIt canvas. A new VisIt visualization will automatically create a new VisIt canvas for visualization and graphical analysis of the mesh data.

The integration of VisIt provides an important extension of the previous Fulcrum visualization capabilities that did not allow for plotting of common 2D and 3D mesh formats such as Exodus, VTK, and the additional formats and features available in VisIt. Figure 9 depicts VisIt embedded in the NEAMS

Workbench with a flexible window layout and visualization of several different formats produced by integrated applications.

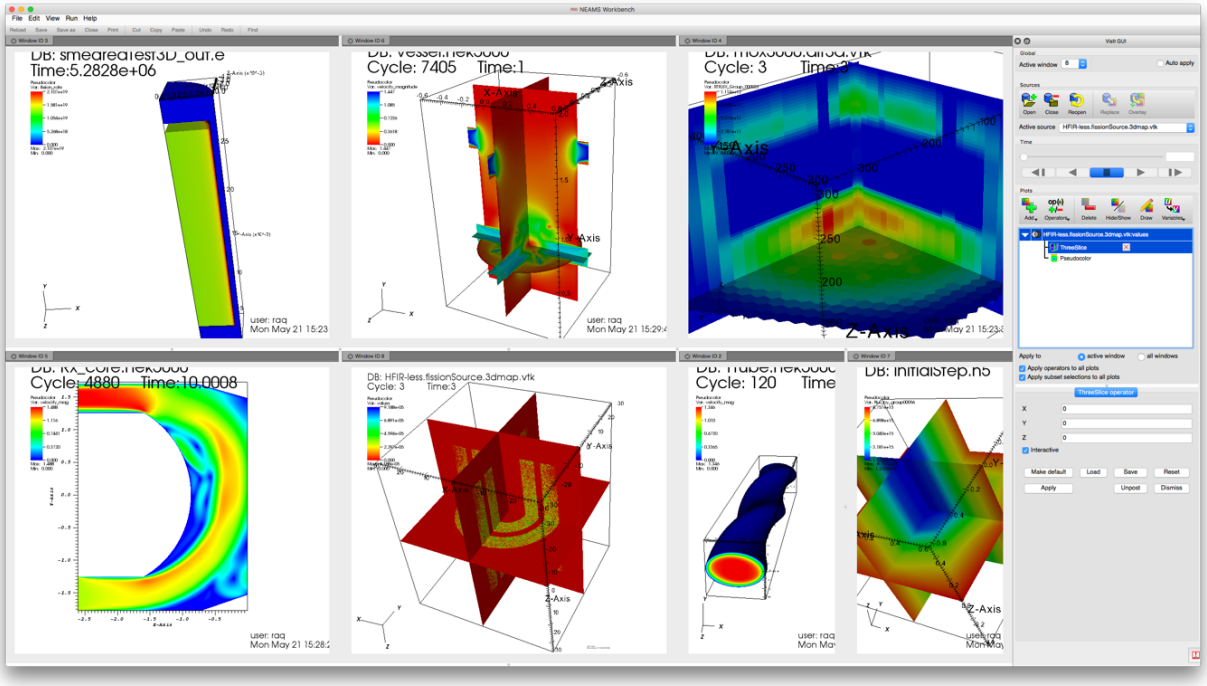

Figure 9. VisIt visualization embedded in the NEAMS Workbench.

An equivalent visualization and analysis capability will be available via the Kitware ParaView tool. Figure 10 illustrates the unreleased Paraview integration currently in testing.

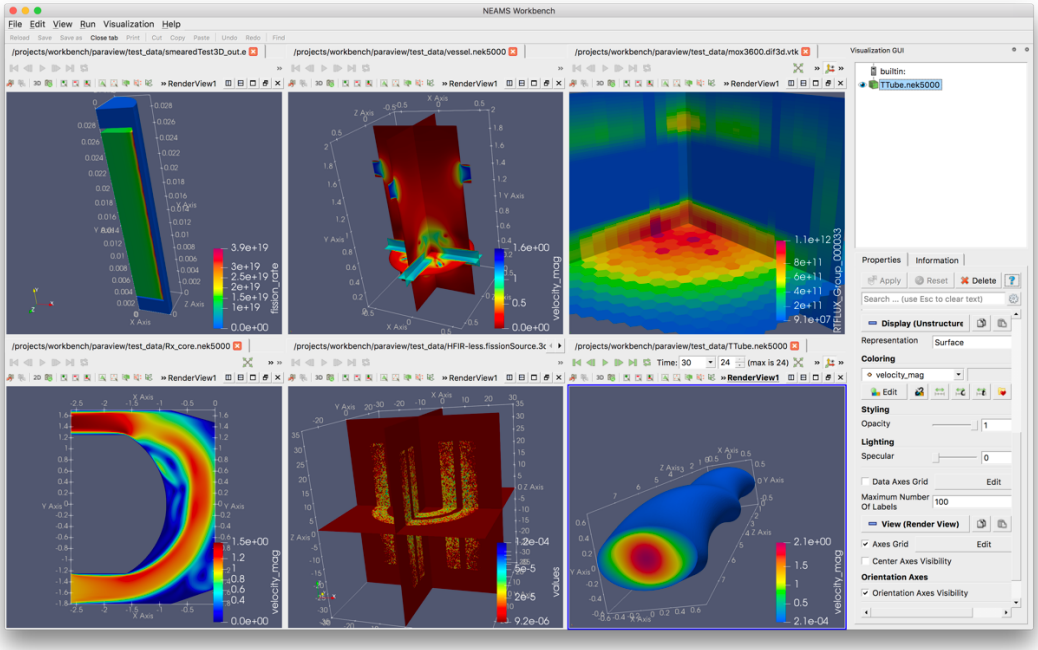

Figure 10. Unreleased Kitware ParaView integration in the NEAMS Workbench

5.2 EXTENDABLE PLOTTING

Because the NEAMS Workbench is intended to support a broad range of codes, any of which may have its own binary or ASCII data formats, an extendable data processor plotting capability was developed to enable the Workbench to extract data using arbitrary processing logic and create plots such as those shown in Figure 9. Any developer or user can create a Workbench processor file that will extract data from a file to create a NEAMS Workbench data plot. The processor file is interpreted at runtime, allowing the user to extend the NEAMS Workbench plotting capabilities for local installation. The processor file can be shared with colleagues, further enabling collaboration.

The Workbench processor files are composed of three components: (1) processor hierarchy, (2) processor filtering, and (3) the processor logic. Because numerous processor files could be associated with different types of analysis, an optional hierarchy feature is available to allow the user to dictate the organization of the processors in the Workbench user interface. This hierarchy allows many processors to be binned into fewer logical groups. In addition to processor organization via hierarchy, the extensions and "filter_pattern" features allow the user to limit the files for which the processor is enabled according to the given file extensions and/or the specified "filter_pattern". The ability to limit when the processor is enabled ensures that the NEAMS Workbench is streamlined for the data under inspection. The processor consists of plot series information and data extraction logic. The data extraction logic uses any command line utility to extract data into spreadsheet format. The *Grep*, *Awk,* and *Python* programs are most commonly used and are available across all supported platforms. The data extraction logic can create multiple series, one per spreadsheet. The plot series can reference the data using a familiar, Excel-like cell-reference mechanism that allows for capturing the keys, values, and uncertainties (low, high). Line style, axis labels, and scale can all be specified. In addition to 2D line, step, and scatter graphs, bar charts and color map plotting are also available. Figure 11 depicts application output plotting enabled by integrated Workbench processors for several integrated applications.

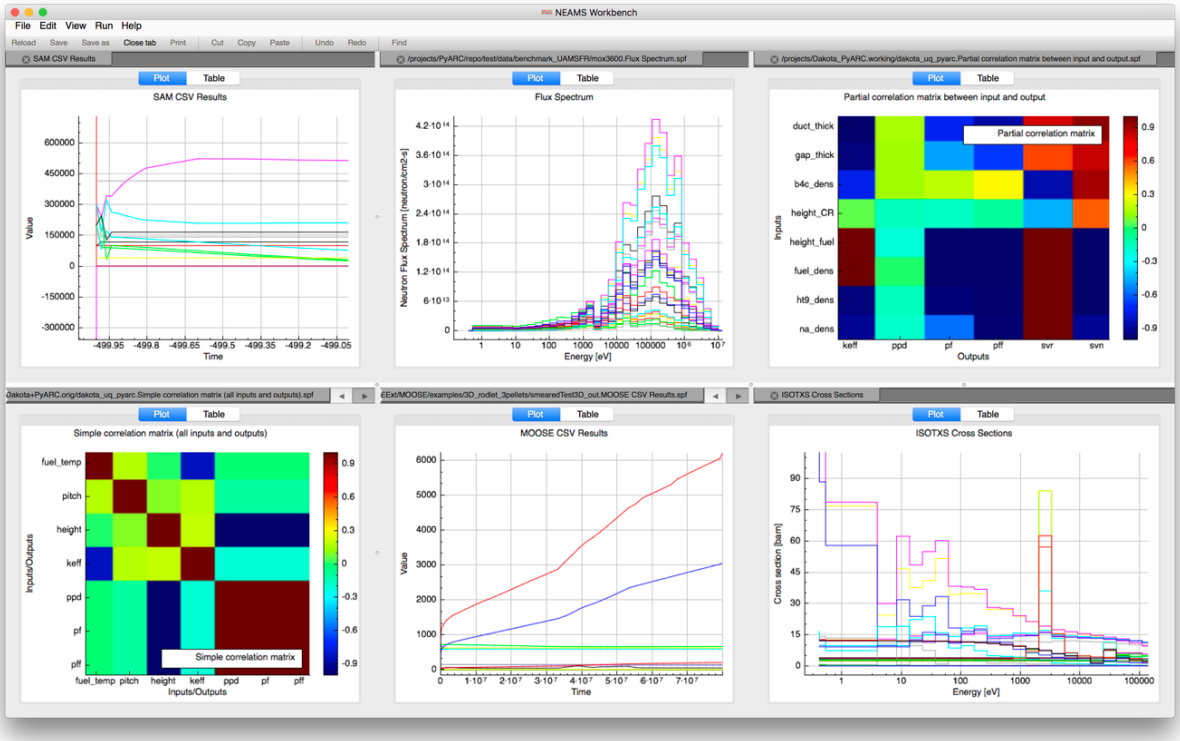

Figure 11. Extendable Workbench processor plotting.

5.3 GEOMETRY VIEWER

Several of the currently integrated applications have geometry input that is describable with combinations of primitive geometry. The NEAMS Workbench has a flexible geometry engine that is also used in the Shift Monte Carlo radiation transport code. The SCALE and ARC geometry descriptions are supported in the NEAMS Workbench geometry viewer (Figure 12).

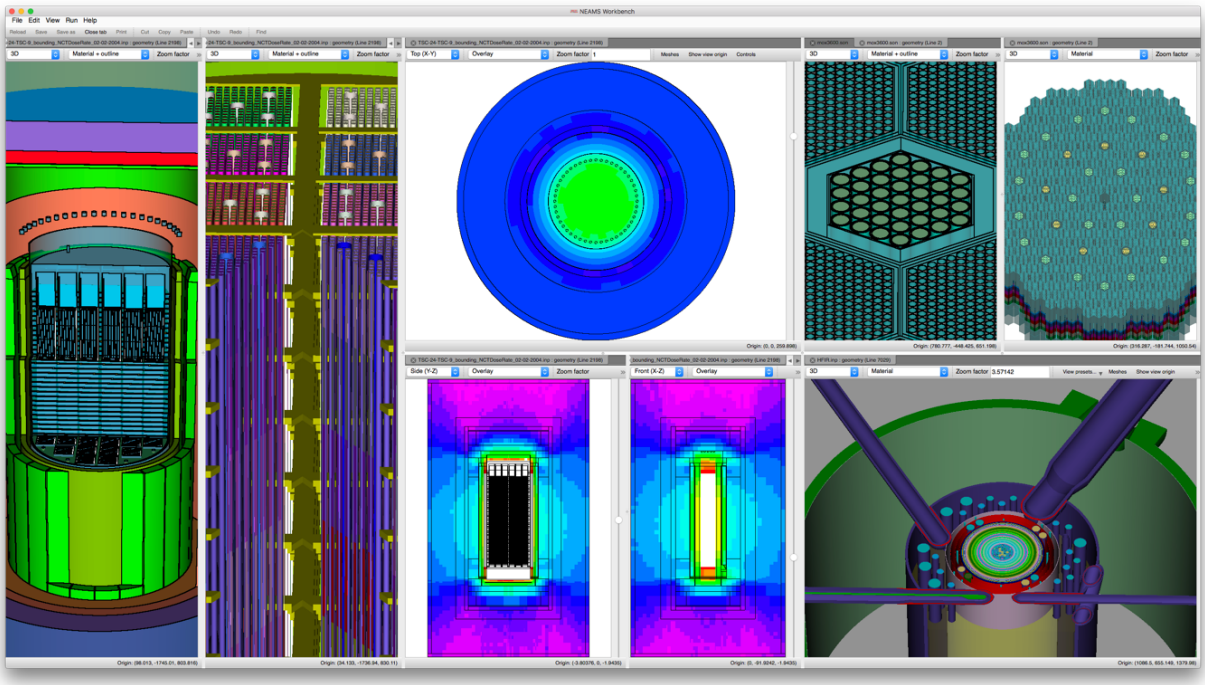

Figure 12. NEAMS Workbench integrated solid geometry viewer.

6. RUNTIME ENVIRONMENT

A goal of the NEAMS Workbench is to facilitate transition from conventional tools to high-fidelity tools. Many codes have different means of launching a calculation, and some have multiple means. A generic runtime environment interface was created to broker application grammar interpretation, input execution, and output, and to provide a consistent interface through which the user and the NEAMS Workbench can interact with each tool in all necessary modes of operation (e.g., serial, parallel, and scheduled execution).

Tools with no runtime environments require the user to manually conduct all steps associated with running them. For example, the user might be required to copy the problem input file into a specific location with a specific file name, such as *input*, and then invoke the application, thus producing temporary scratch files and output file(s). If multiple simultaneous calculations are desired, then many complications follow that will likely lead to failed or erroneous results. Other codes are distributed with sophisticated job management capabilities designed for use in quality-assured licensing calculations that should not be disrupted.

As such, the NEAMS Workbench enables the integration of customized runtime environments for each integrated tool. A runtime script provides the setup, execution, or finalization logic needed to fulfill the runtime interface. The setup logic might create a working directory, *TMPDIR*, and then copy the *problem.inp* into the *TMPDIR* as *TMPDIR/input*. The execution might invoke the application executable, passing application messages back to the calling application (e.g., command console, Workbench). The finalized logic might (1) combine the output files located in *TMPDIR* into logical order, (2) copy the output back into *problem.out*, residing next to *problem.inp*, and (3) delete the *TMPDIR* to clean up after itself. Once integrated by a developer or expert users, runtime scripts are accessible through a dropdown menu in Workbench. The scripts are written in Python and stored in the runtime environment (*rte*) directory inside Workbench, so they can be customized and supplemented as tools and use scenarios evolve.

A Workbench runtime script can be developed to allow consistent invocation for any specific application logic, and it may provide great convenience relative to the application's typical command-line interface. The runtime's grammar interface enables the NEAMS Workbench to allow the user to automatically retrieve an application's grammar information. This allows users to switch between different versions of integrated applications (e.g., Dakota 6.5, 6.10), and it allows developers to update their applications and have the NEAMS Workbench provide the version-specific input parameters and checks. A special runtime—setup_remote—is available for remote application configuration and job submission. The setup_remote runtime generates a remote application runtime. This has been tested with portable batch system (PBS)-type scheduling and is intended to be compatible with other popular schedulers (SLURM, etc.). As the runtime environment matures and additional features like a workflow management are added for the base class, all incorporated runtimes will benefit. The NEAMS Workbench runtime environment allows all integrated codes to have jobs launched locally from the user's desktop or laptop, and remotely, as shown in Figure 13. The remote job launch can be against a single Linux box (i.e., a machine with no scheduler but with available cores) or an industrial cluster with a scheduler. This highlights the ability for Windows users to conduct job creation, launch, and analysis tasks from a familiar environment while using a code that is not available in the Windows operating system.

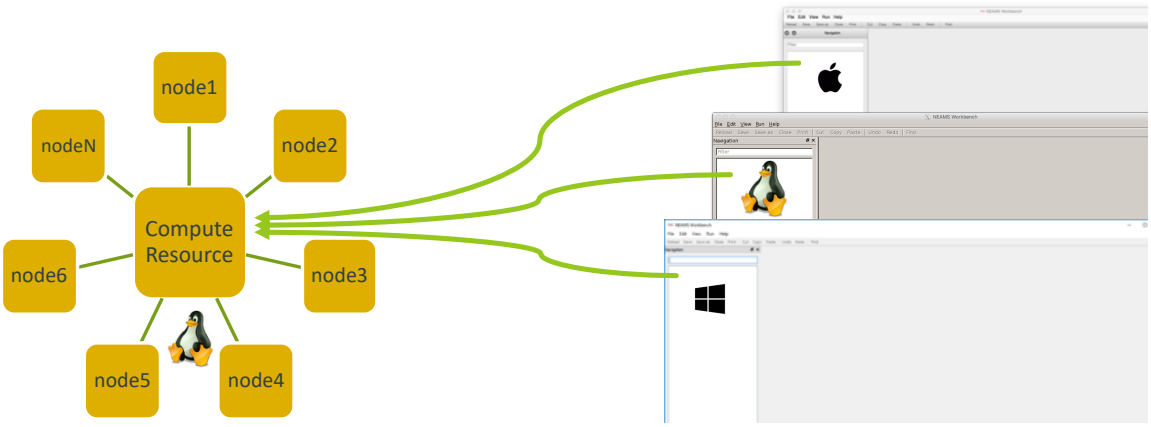

Figure 13. NEAMS Workbench OS-independent remote job launch on Linux compute resource.

7. CODE INTEGRATION

To facilitate the use of NEAMS-developed codes and many other commonly used production capabilities, a wide range of tools from many teams was integrated for the latest beta release of the NEAMS Workbench. Integration of a new code is enabled by first ensuring that WASP can process its input format. For modern tools that use common input structures, this first step is fairly simple. For legacy codes that use custom-developed unstructured input formats, new custom features in WASP must be developed. Once the input can be read into Workbench in a hierarchical format, the HIVE schema files are developed or generated, generally by following the user documentation to support various blocks of input, confine input to allowed values, and manage interdependencies between multiple input parameters. Next, a runtime environment script is developed that includes routines to properly manage the application grammar, arrange the input files and executables, and retrieve the output data. HALITE templates are developed and added to the Workbench configuration to provide auto-completion of all or part of an application's input. At any time, a software developer or end user can customize or supplement the HIVE schema files or HALITE templates for codes with input formats supported by WASP. The codes integrated for the latest NEAMS Workbench Beta release are detailed below.

7.1 MOOSE APPLICATIONS

The MOOSE framework provides the foundation for many NEAMS developed tools, most visibly the BISON fuels performance code. Initial efforts for the NEAMS Workbench focused on convenient coupling between the MOOSE framework and the NEAMS Workbench. With its modern software design, MOOSE includes a common input format and a common engine for processing input for any MOOSE application. WASP was updated to support the MOOSE input format as one of its known input styles, and MOOSE was updated to generate a HIVE schema file for any MOOSE application. Running a MOOSE application with the command line argument `--definitions` will generate a HIVE schema file representing the current input features of the MOOSE application. This schema is managed by the application's NEAMS Workbench runtime. A generic MOOSE runtime is provided with the NEAMS Workbench. A custom runtime environment must only be developed for each MOOSE application if the command line arguments are not already available in the MOOSE Workbench generic runtime. The existing integration supports all applications, but only the BISON fuel performance and the system analysis module (SAM) [15] have been tested. After running a MOOSE application calculation, the output file can be viewed, and the mesh data files can be visualized with the embedded VisIt tools, as shown in Figure 14 for a BISON fuel performance calculation and Figure 15 for a SAM calculation.

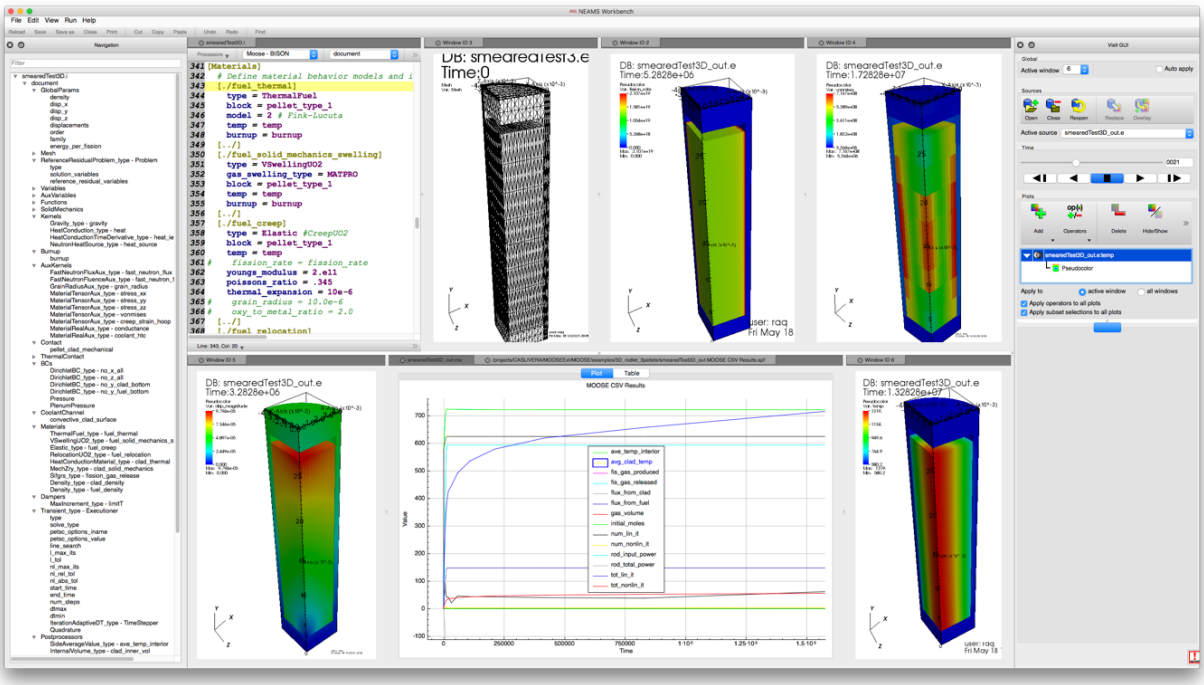

Figure 14. BISON fuel performance integration in the NEAMS Workbench.

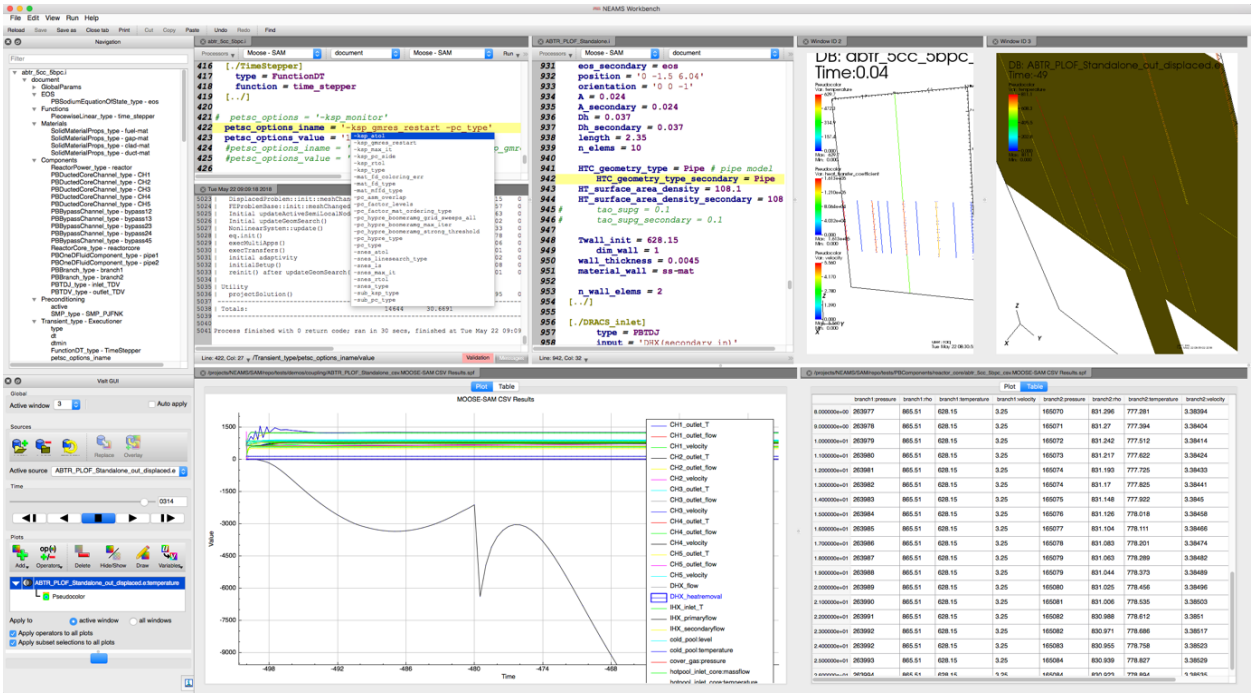

Figure 15. SAM integration in the NEAMS Workbench.

7.2 ARGONNE REACTOR COMPUTATION SUITE

The traditional Argonne Reactor Computation suite (ARC) for fast reactor analysis has been integrated into the NEAMS Workbench at the request of the Advanced Reactor Technologies team members who routinely use these tools. In this work, a new common input was developed using the Standard Object Notation input format developed for SCALE 6.2 and was revised in WASP. This common input is used to run calculations with MC^2-3, DIF3D, REBUS, and PERSENT for integrated, problem-dependent, cross section preparation, core analysis, depletion, and sensitivity/uncertainty analysis. Before the common input was developed, each of these calculations required a separate input file, and many of the input formats were so difficult to use that a specialized script was often required to generate them. With the ARC/Workbench integration creating the PyARC [16] Workflow manager, users can easily create engineering-style input, have the Workbench generate the input for the codes, launch the calculations using a customized runtime environment, and visualize the results with the embedded Workbench processor files and VisIt tool. An integrated ARC/Workbench input and associated workflow are shown in Figure 16. An example ARC calculation in the NEAMS Workbench is shown in Figure 17. Users who access the ARC codes through the Workbench will have easier access to the advanced codes of NEAMS, which will be able to leverage common input parameters in the future, especially with the integration of MCNP® and PROTEUS.

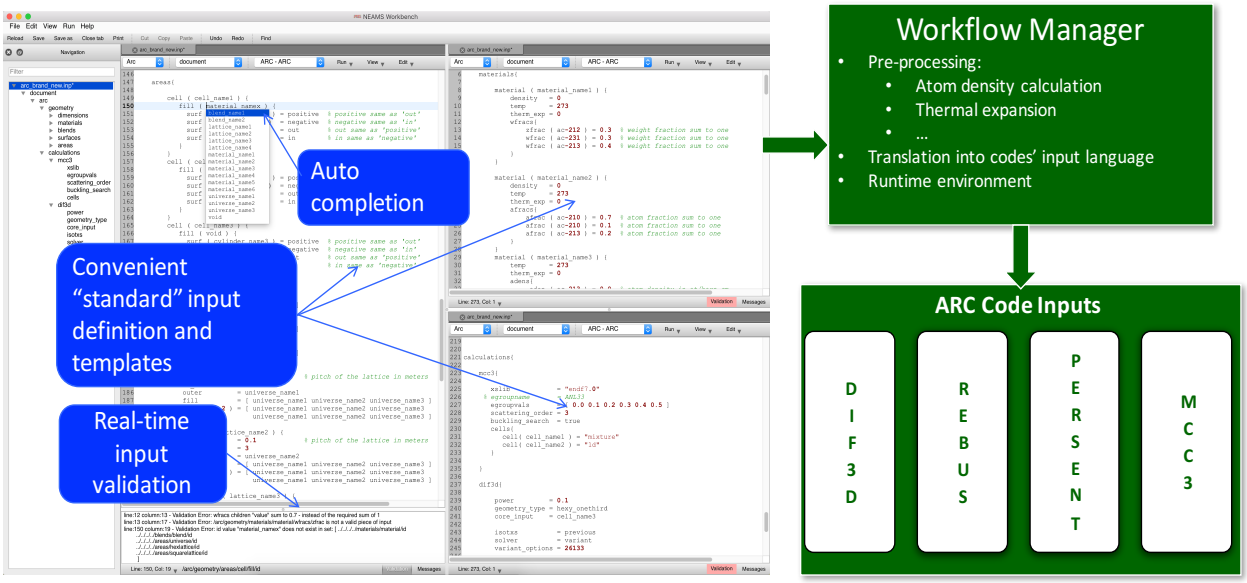

Figure 16. Prototype ARC/Workbench integration and associated workflow manager.

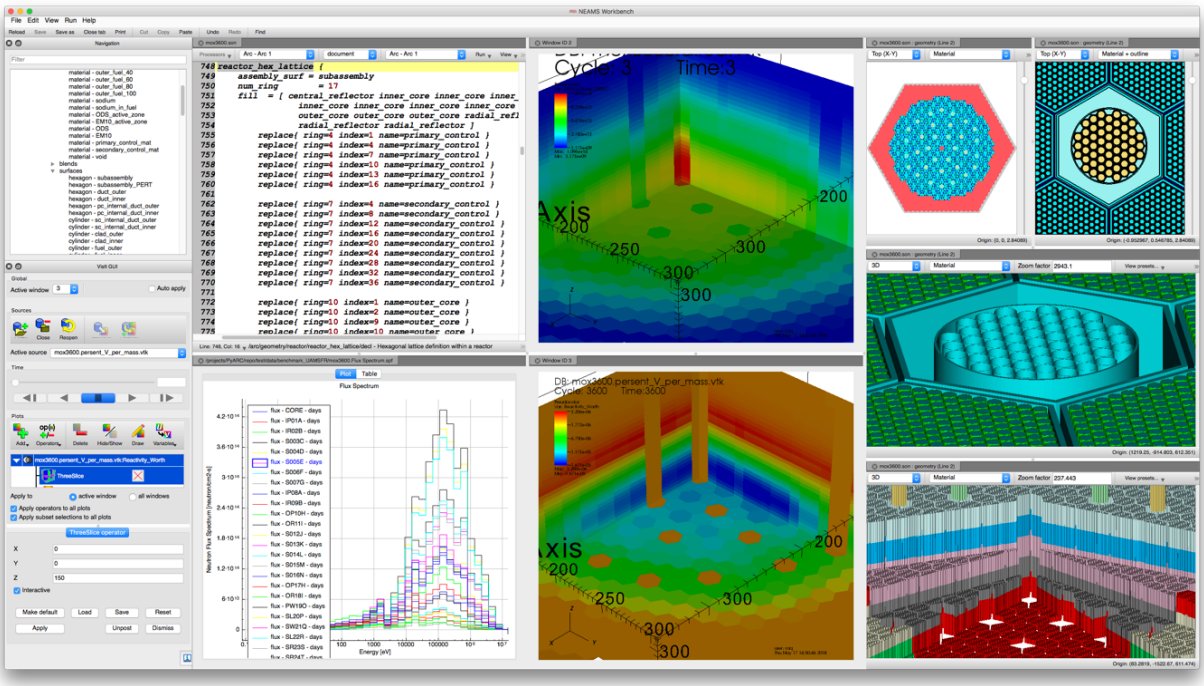

Figure 17. ARC neutronics fast reactor integration in the NEAMS Workbench.

7.3 DAKOTA UNCERTAINTY QUANTIFICATION AND OPTIMIZATION

The Dakota suite of iterative mathematical and statistical methods has been integrated into the NEAMS Workbench. The suite is from Sandia National Laboratories, and the methods interface with many computational models. A new definition-driven input interpreter was developed and added to WASP to support the Dakota input format, and many updates were implemented in the native Dakota schema file, dakota.xml, to provide for improved consistency in the Dakota input. A Python translation script translates the dakota.xml into a Workbench schema file. A customized runtime environment and plotting capabilities allow the user to conveniently run and visualize extracted data from the output file. An example calculation of Dakota conducting uncertainty quantification with the MOOSE BISON fuel performance application in the NEAMS Workbench are shown in Figure 18. A demonstration of the coupling between Dakota and the ARC suite of codes in the NEAMS Workbench, is shown in Figure 19 [17].

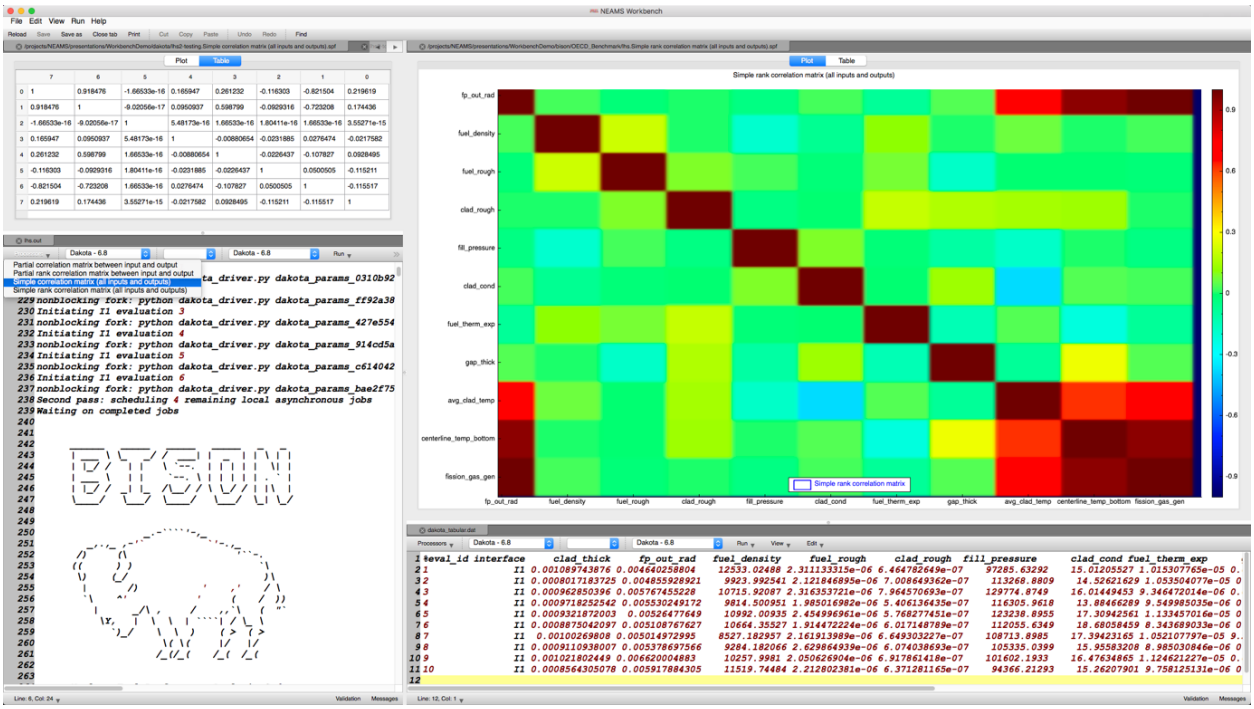

Figure 18. Dakota integration running MOOSE BISON in the NEAMS Workbench.

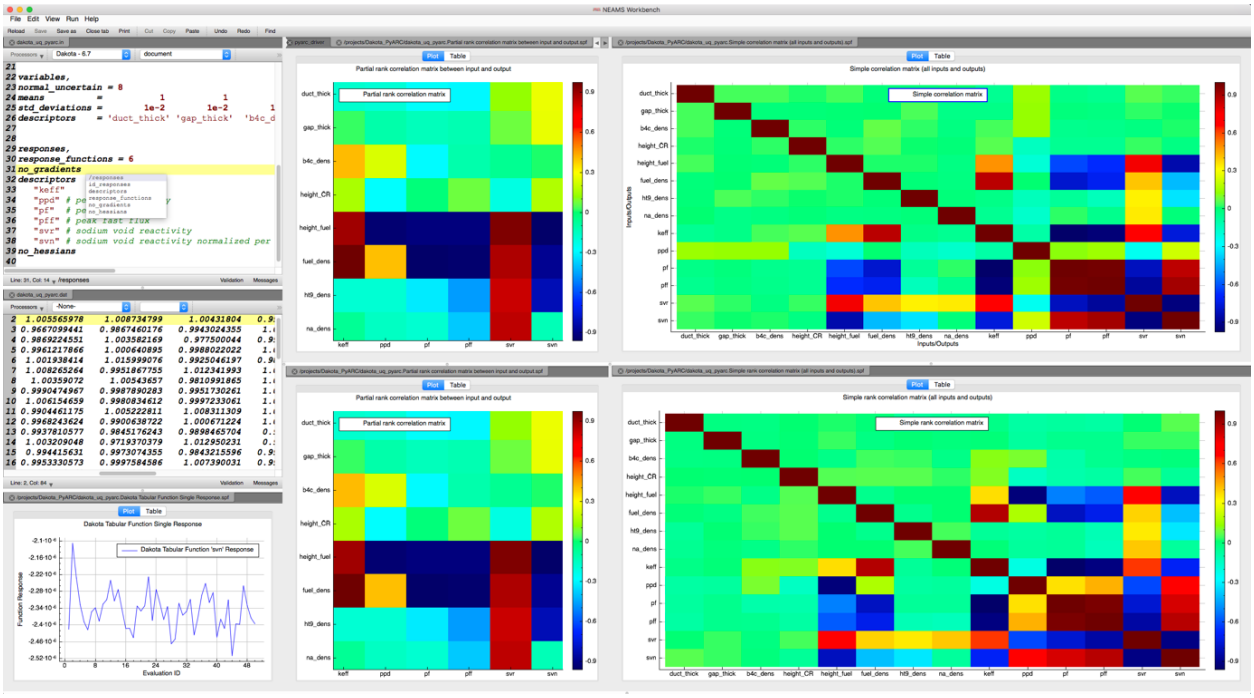

Figure 19. Dakota integration running ARC neutronics fast reactor tools in the NEAMS Workbench.

7.4 SCALE CODE SYSTEM

Because the NEAMS Workbench is an evolution of the Fulcrum user interface from the SCALE code system, it supports the dozens of verified and validated design and licensing tools used for criticality safety, reactor physics, radiation shielding, radioactive source-term characterization, and sensitivity/uncertainty analysis for a full range of systems—light-water reactors (LWRs), advanced reactors, and research/test reactors. Examples of SCALE integration are shown in Figure 3 through Figure 6 above and in Figure 20 below.

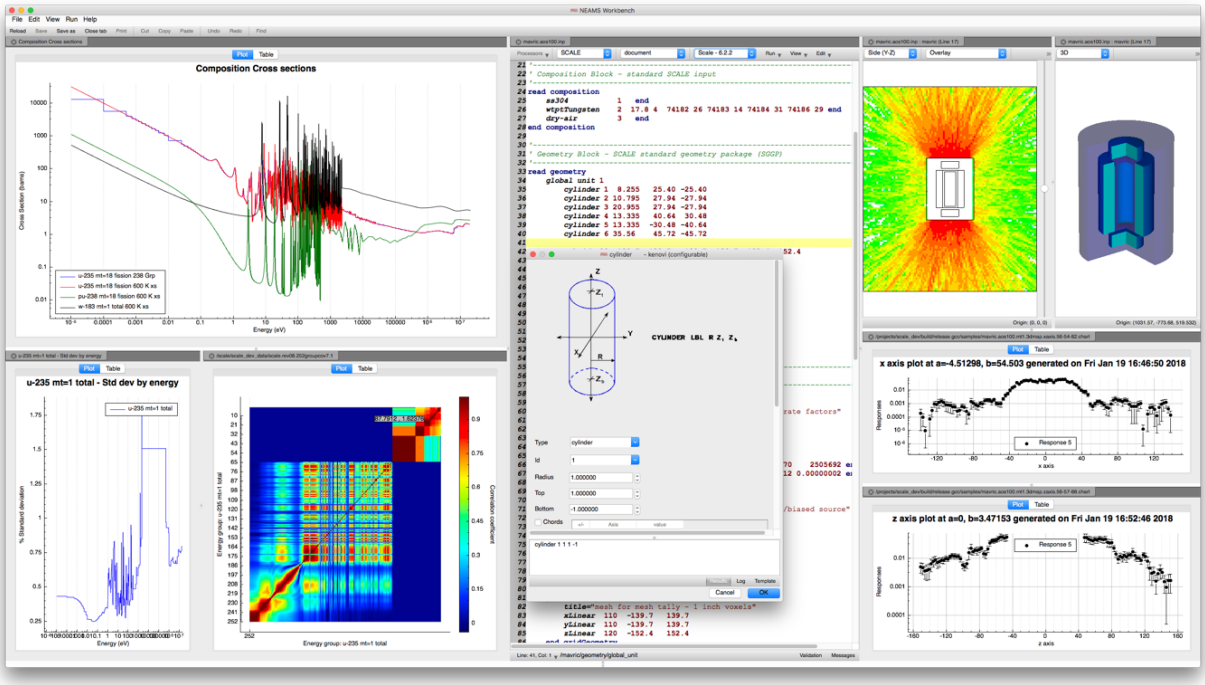

Figure 20. SCALE code system in the NEAMS Workbench.

7.5 MCNP® PARTICLE TRANSPORT

The initial integration of MCNP® into the NEAMS Workbench has been completed in collaboration with Rensselaer Polytechnic Institute (RPI) [18] and provides input verification and autocompletion via an XText-generated language server [19]. In addition to input processing, autocompletion, and verification, the NEAMS Workbench supports MCNP® job execution on local and remote resources. Geometry cannot be visualized at this stage. Figure 21 illustrates this initial integration, which is available for download from https://code.ornl.gov/neams-workbench/downloads/tree/beta-mcnp-support.

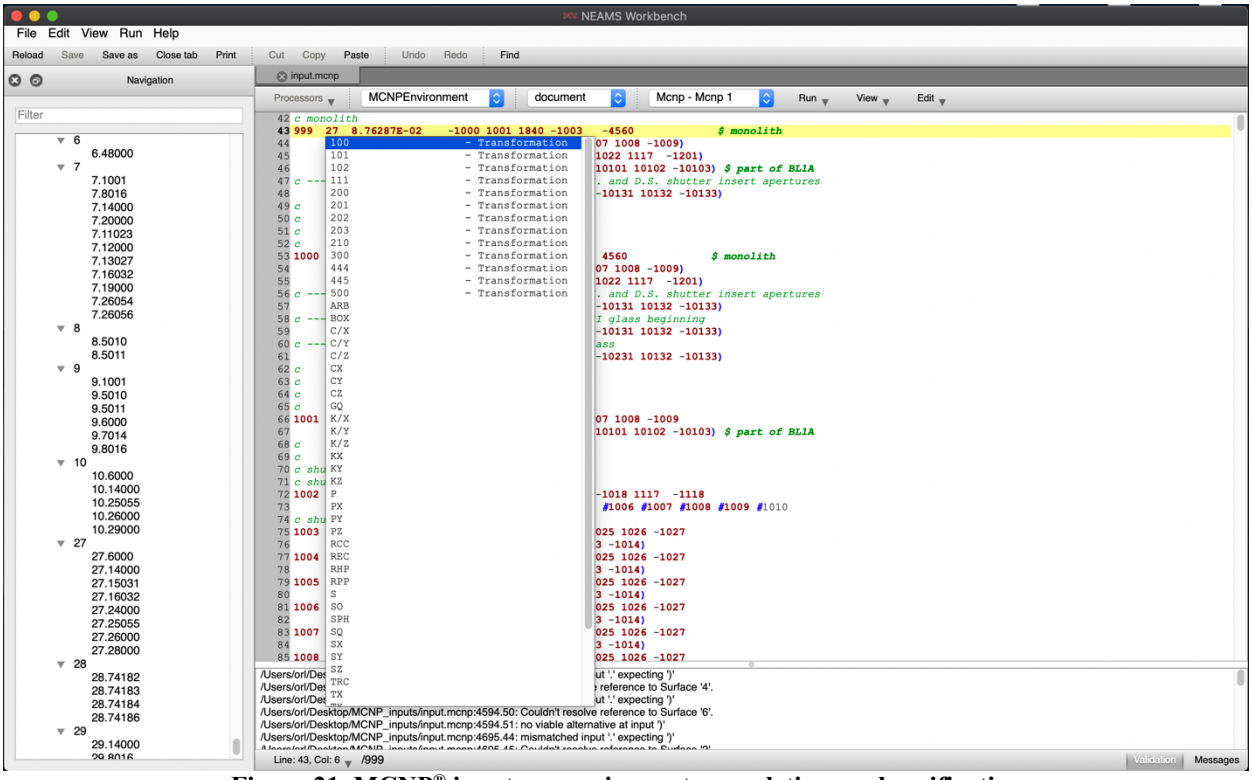

Figure 21. MCNP® input processing, autocompletion, and verification integrated into the NEAMS Workbench.

7.6 Nek5000 COMPUTATIONAL FLUID DYNAMICS

The Nek5000 computational fluid dynamics (CFD) code from the SHARP Multiphysics framework provides high-order methods to solve CFD problems. It is designed to scale from laptops to supercomputers. The user workflow for Nek5000 can be quite involved, requiring the user to manager four session files and conduct several preprocessing steps, including compilation of the Nek5000 executable itself. Once the preprocessor steps are complete, the Nek5000 session can be executed. Conducting postprocessing involves an additional step of generating the Nek5000 output for visualization by VisIt. Figure 22 depicts the workflow.

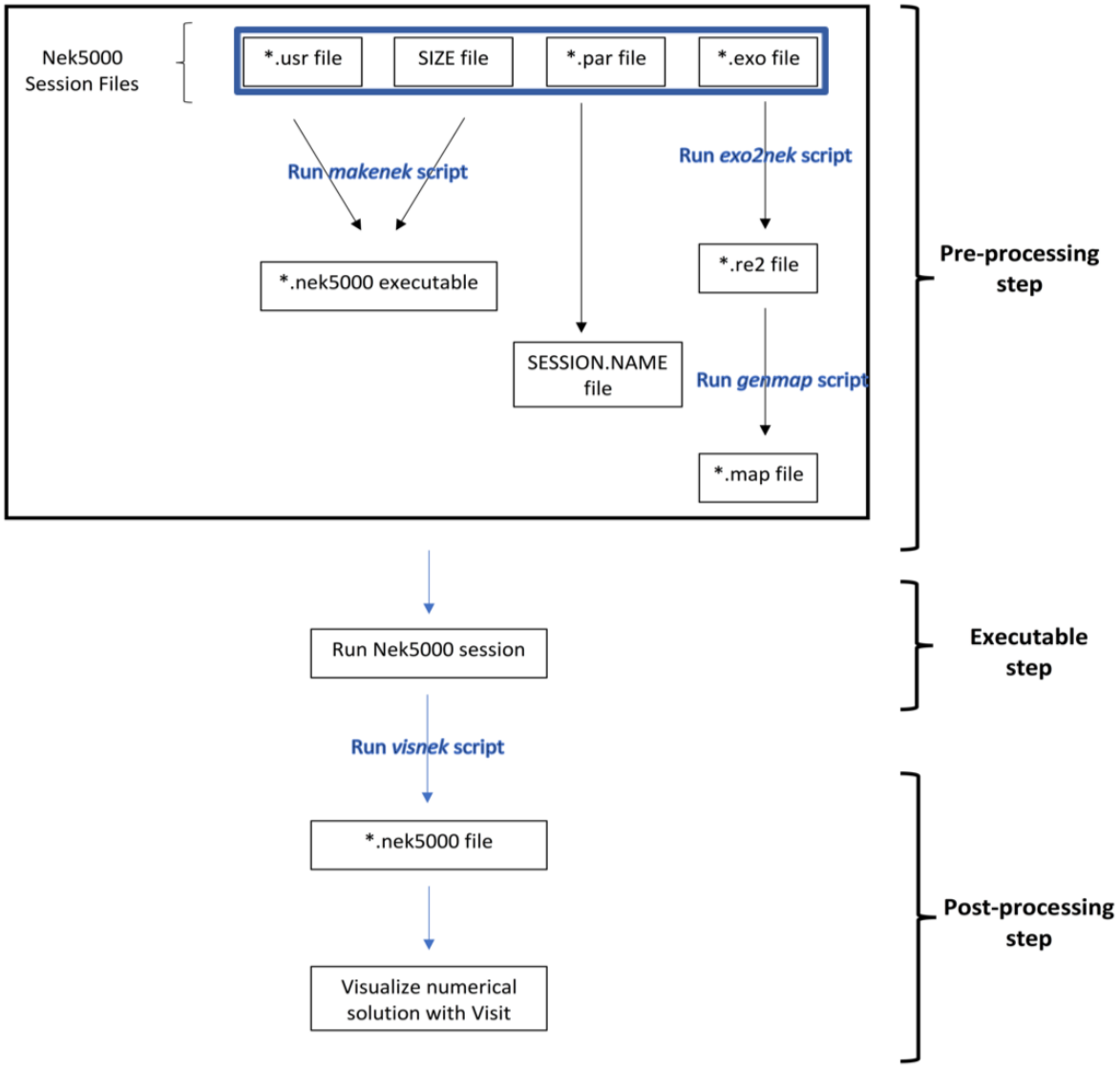

Figure 22. Typical Nek5000 workflow.

The NEAMS Workbench integration streamlines this workflow with the introduction of the Nek5000 Workbench runtime [20]. The WASP package required a minor update to support the Nek5000 PAR input format to support Nek5000 PAR input parsing. An input schema was developed to enable input checks and auto-completion. The input was supplemented with a runtime block containing parameters that allow the user to control the Nek5000 runtime's preprocessing and postprocessing steps. Figure 23 depicts Nek5000 input and auto-completion and VisIt mesh and results visualization in the NEAMS Workbench.

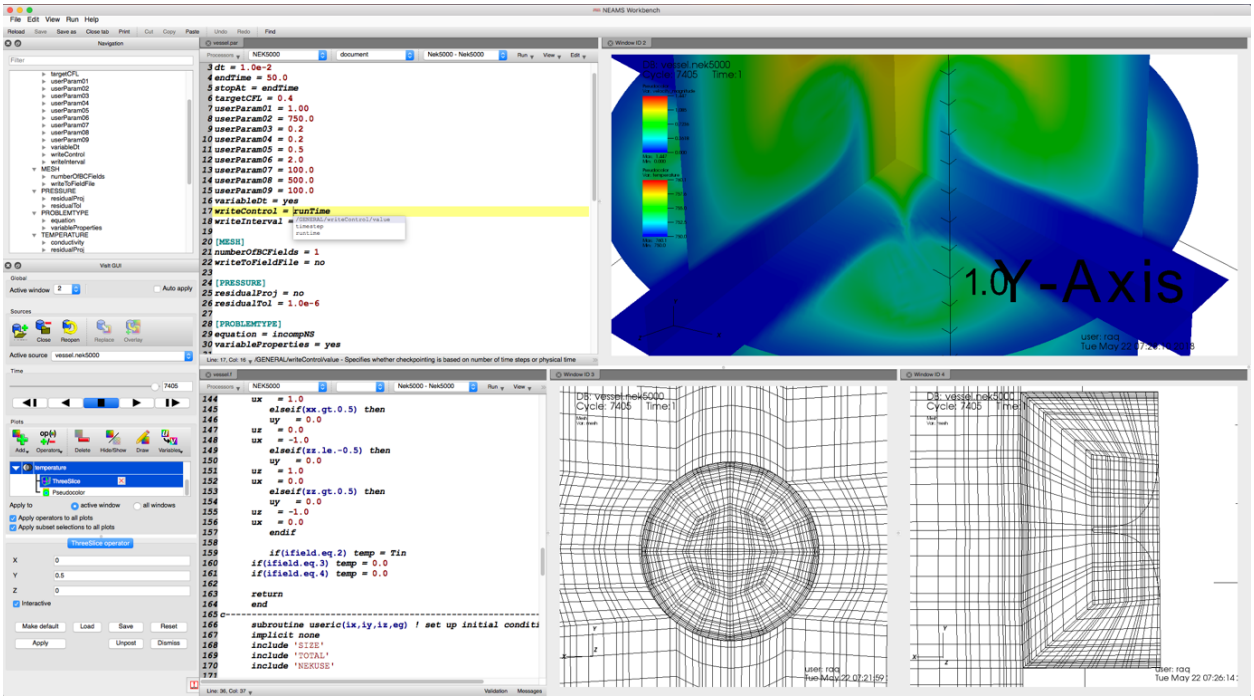

Figure 23. Nek5000 CFD integrated in the NEAMS Workbench.

While the Nek5000 Workbench runtime and input parameters greatly streamline user interaction, the user is still required to manage four session files, which are prone to error. The Nek5000 for nuclear, or Nek4Nuc, Workbench runtime was developed to further streamline the Nek5000 workflow and limit duplicate, error-prone, session file data. Because of the WASP input and templating capabilities, the Nek4Nuc Workbench runtime was easily able to incorporate the additional user data as additional input parameters in the Nek5000 PAR format extension PARN. This input extension and the workflow extension facilitated by the HALITE templates removed the requirement for two session files and the associated duplicate session data. As depicted by the red box in Figure 24 . the Nek4Nuc workflow enables the user to interact with only the essential, required, Nek5000 user data.

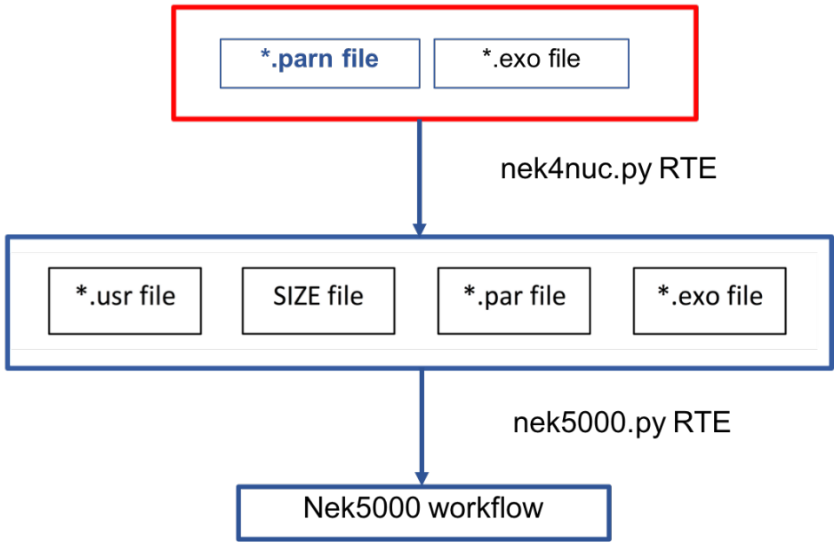

Figure 24. Nek4Nuc CFD integrated in the NEAMS Workbench.

7.7 CTF SUBCHANNEL THERMAL-HYDRAULICS

The CTF subchannel thermal-hydraulics code has been developed for LWR analysis and is an important part of the CASL VERA tool [21]. CTF has been integrated via the SubKit [22] interface, which utilizes the WASP Definition-Driven Interpreter (DDI) package for input processing. This facilitates the integration of CTF-SubKit input processing, autocompletion, and verification in the Workbench. CTF VTK results are integrated via the VisIt visualization toolkit; the HDF5 output integration is in progress. CTF-SubKit's integration into Workbench (Figure 25) streamlines the input creation, job launch, results visualization, and analysis, and it significantly reduces the CTF getting-started learning curve.

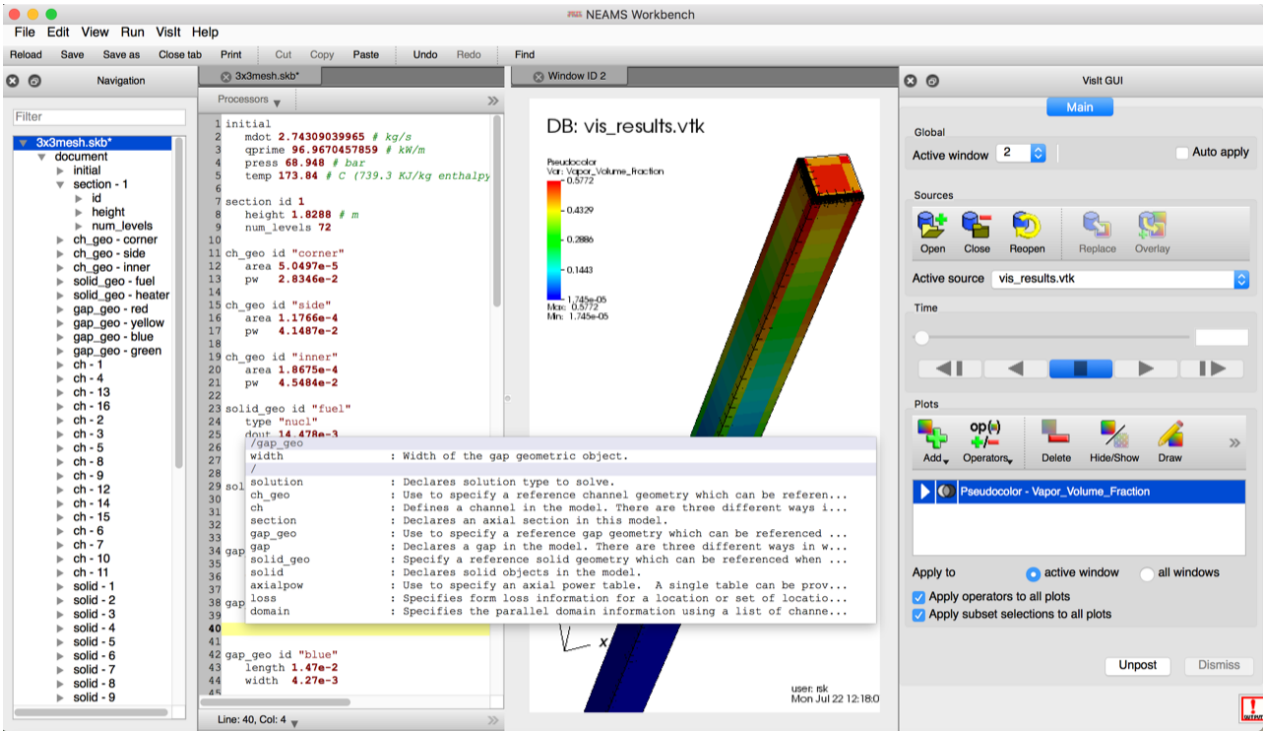

Figure 25. CTF Subchannel TH integrated via the SubKit interface into the NEAMS Workbench.

7.8 CTFFUEL LWR FUEL ANALYSIS

The CTFFuel is an LWR fuel rod modeling capability contained in CTF. It is extracted into a convenient interface specifically to fulfill the needs of CASL and fuel analysts [23]. The input syntax for CTFFuel is supported by WASP's VERA Input Interpreter (VII), making integration of input processing straightforward (Figure 26). CTFFuel VTK results are available for visualization and analysis through the integrated VisIt visualization toolkit. The HDF5 output integration is in progress at this writing.

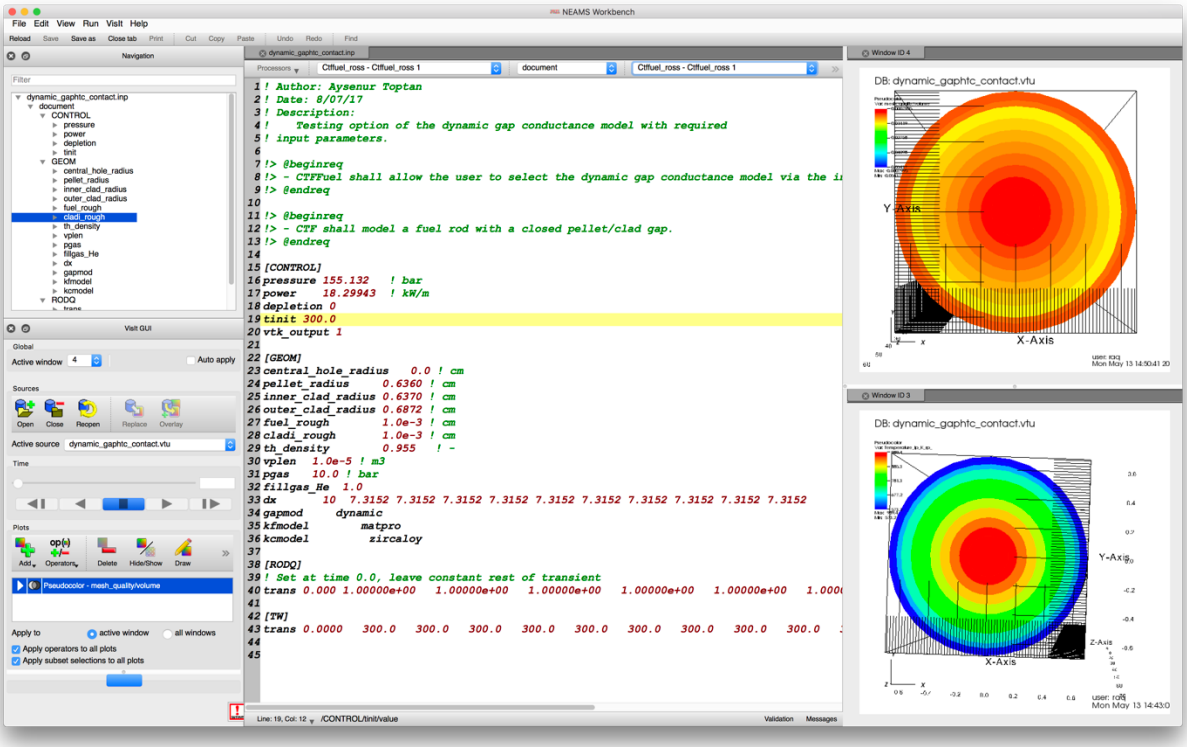

Figure 26. CTFFuel fuel analysis capability integrated into the NEAMS Workbench.

7.9 CASL VERA

The VERA integration is in progress. The remaining tasks involving job launch, VERA-View/Out, and the advanced VERA-In defaults mechanism. Current support for VERA's SILO-formatted output and input processing, autocompletion, and verification are depicted in Figure 27. The current status provides some useful capabilities in input creation and navigation, but it does not yet fully enable users in all capabilities due to the inability to launch VERA jobs on local or remote compute resources and the lack of full output integration. The VERA-In default mechanism provides for file-includes. The file-include mechanism is supported for files located relative to the input, but it does not support file-includes relative to the install of VERA. This is in part due to VERA potential to be a remote installation.

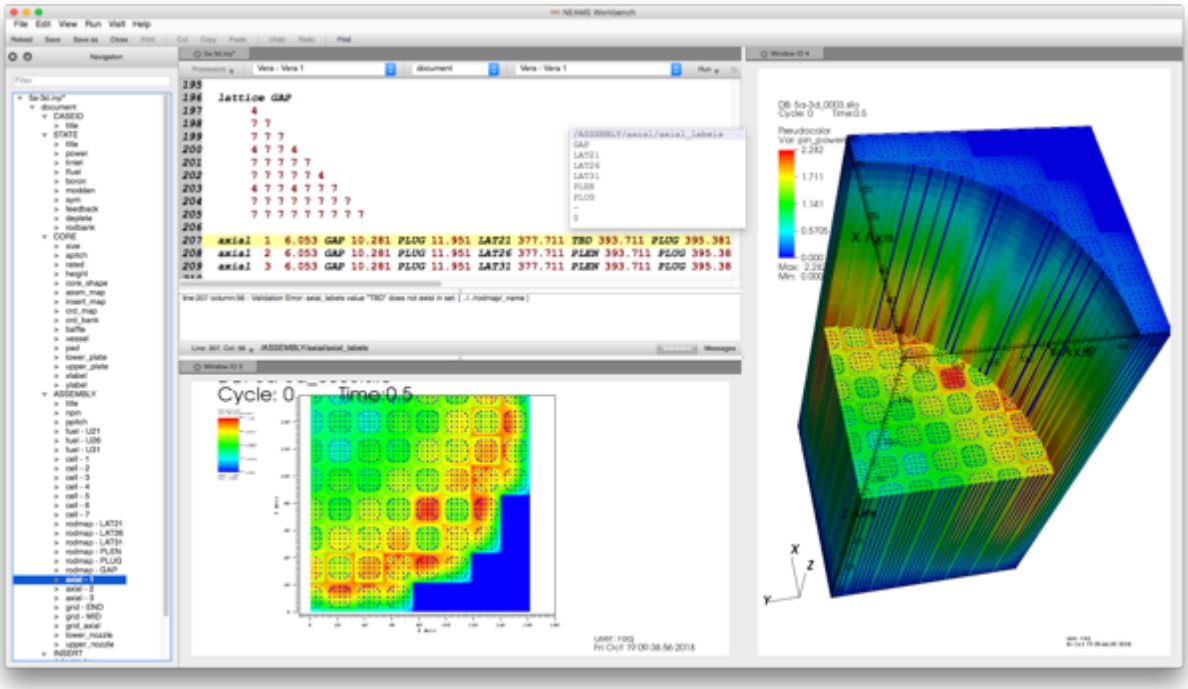

Figure 27. Preliminary VERA input and results visualization integration in the NEAMS Workbench.

7.10 CODE INTEGRATION SUMMARY

Table 1 includes the codes that are integrated for the NEAMS Workbench Beta release, as well as near-term code integration activities. Under a Nuclear Energy University Program awards, RPI is partnering with ORNL's Workbench team, ANL's PROTEUS team, the Los Alamos National Laboratory (LANL) MCNP® team, and North Carolina State University (NCSU) to integrate these tools.

Table 1. Code integration for the NEAMS Workbench

Tool	Application	Status	Integration lead
ARC	Fast reactor analysis	Beta	ANL
BISON	Fuel performance	Beta	ORNL/INL
Dakota	Uncertainty quantification and model optimization	Beta	SNL/ORNL
MOOSE	General purpose multiphysics framework	Beta	ORNL/INL
NEK5000	Computational fluid dynamics analysis	Beta	ORNL
SAM	MOOSE tool for single phase systems analysis	Beta	ORNL/ANL
SCALE	Widely used multipurpose neutronics and shielding analysis	Beta	ORNL
VERA-CS	CASL multiphysics core simulator tools	In progress	ORNL
MCNP®	Widely used Monte Carlo radiation transport code	In progress	RPI/LANL
PROTEUS	Three-dimensional unstructured grid finite element neutron transport solver	In progress	RPI/ANL
CTF	Subchannel thermal-hydraulics tool	Beta	NCSU/ORNL
CTFFuel	LWR fuel analysis tool	Beta	NCSU/ORNL

8. AVAILABILITY

The NEAMS Workbench Beta is available to interested users and developers. An open source licensed precompiled binary distribution is available at https://code.ornl.gov/neams-workbench/downloads. The user documentation and additional information for operating many Workbench features and integrating applications is provided under the `Help` drop-down.

If any questions arise or if any issues occur, please email nwb-help@ornl.gov for support.

9. CONCLUSIONS

The NEAMS Workbench is a new initiative created in response to the needs of the design and analysis communities. It is intended to enable end users to apply high-fidelity simulations to inform lower order models for the design, analysis, and licensing of advanced nuclear systems. In its beta release capacity, it enables enhanced input editing, validation, and navigation capabilities to assist new users of NEAMS tools in getting started. In addition, the NEAMS Workbench provides basic job execution via an extensible runtime environment and convenient output visualization and analysis capability via fast 2D data plotting, as well as 2D and 3D mesh analysis and visualization through the integrated VisIt toolkit.

The NEAMS Workbench will facilitate the transition from conventional tools to high-fidelity tools by providing a common user interface and common user input processing capabilities: flexible input formats, a hierarchical validation engine, and a template engine. An extensible runtime environment ensures that computational environments can be considered. Integrated applications and associated system templates will continue to broaden the NEAMS tools user community and will facilitate system analysis and design.

The open source licensed binary release of the NEAMS Workbench user interface and common input processing capabilities will further enable future collaboration and will enable extensions of NEAMS Workbench for proprietary industry application modeling and simulation needs.

10. ACKNOWLEDGMENTS

This research was sponsored by the US Department of Energy's Nuclear Energy Advanced Modeling and Simulation program. Argonne National Laboratory's work was supported by the US Department of Energy under contract DE-AC02-06CH11357. Additionally, special thanks are in order for the collaborators who made contributions or provided direction that facilitated application integration. Specifically, thanks to Cody Permann and Brian Algers for their guidance and assistance with integration of the MOOSE framework applications; Rui Hu for his guidance and assistance with integration of SAM; Laura Swiler, Brian Adams, and Elliot Ridgway for their guidance and assistance with integration of the Dakota; Nicolas Stauff for his collaboration and guidance on ANL's ARC integration and creation of the PyARC coupling module; Harinarayan Krishnan for his guidance and contributions to Workbench for the VisIt visualization toolkit integration; Srdjan Simunovic for his assistance with VERA-In; Bob Salko and Vineet Kumar for collaboration and integration of CTF via the SubKit interface; Bob Salko, Agustin Abarca, and Maria Avramova for CTFFuel collaboration and integration; Wei Ji, Kurt Dominesey, Peter Kowal, Mathieu Dupon, Forest Brown, Matthew Eklund and Jonathan Eugenio for collaboration and integration of MCNP®; and Bob O'Bara, T.J. Corona, Ben Boeckel, and Utkarsh Ayachit for collaboration and contributions to Workbench for ParaView visualization toolkit integration. Additional thanks to Kaylee Cunningham for valuable testing and feedback for MOOSE Bison.

11. REFERENCES

[1] Gaston, D. R., C. J. Permann, J. W. Peterson, A. E. Slaughter, D. Andrš, Y. Wang, et al. 2014. "Physics-based multiscale coupling for full core nuclear reactor simulation." *Ann. Nucl. Energy* 84: 45–54.

[2] Williamson, R. L., J. D. Hales, S. R. Novascone, M. R. Tonks, D. R. Gaston, C. J. Permann, D. Anders, and R. C. Martineau. 2012. "Multidimensional multiphysics simulation of nuclear fuel behavior." *J. Nucl. Mater.* 423: 149–163.

[3] Yu, Y. Q., E. R. Shemon, J. W. Thomas, Vijay S. Mahadevan, Ronald O. Rahaman, and Jerome Solberg. 2016. *SHARP User Manual.* ANL-NE-16/6. Argonne, Illinois: Argonne National Laboratory.

[4] Rearden, B. T., R. A. Lefebvre, B. R. Langley, A. B. Thompson, and J. P. Lefebvre. 2018. *NEAMS Workbench 1.0 Beta.* ORNL/TM-2018/752. Oak Ridge, Tennessee: Oak Ridge National Laboratory.

[5] Lefebvre, R. A., et al. 2018. *NEAMS Workbench Status and Capabilities.* ORNL/TM-2018/992. Oak Ridge, Tennessee: Oak Ridge National Laboratory.

[6] ANL. 2014. *ARC 11.0: Code System for Analysis of Nuclear Reactors.* Argonne, Illinois: Argonne National Laboratory. Available from Radiation Safety Information Computational Center as CCC-824.

[7] Rearden, B. T., and M. A. Jessee, Eds. 2016. *SCALE Code System.* ORNL/TM-2005/39, Version 6.2. Oak Ridge, Tennessee: Oak Ridge National Laboratory. Available from Radiation Safety Information Computational Center as CCC-834.

[8] Pelowitz, D. B., Ed. 2013. *MCNP6 User's Manual, Version 1.0.* LA-CP-13-00634, Rev. 0, May. Los Alamos, New Mexico: Los Alamos National Laboratory. Available from Radiation Safety Information Computational Center as CCC-810.

[9] Turner, J. A., K. Clarno, M. Sieger, R. Bartlett, B. Collins, R. Pawlowski, R. Schmidt, and R. Summers. 2016. "The Virtual Environment for Reactor Applications (VERA): design and architecture." *J. Comput. Phys.* 326: 544–568.

[10] LLNL. *VisIt Visualization Tool (2002–2016).* Livermore, California: Lawrence Livermore National Laboratory. https://wci.llnl.gov/codes/visit.

[11] Kitware. *ParaView* (2002–2016). Clifton Park, New York: Kitware, Inc. http://www.paraview.org.

[12] Lefebvre, R. A., B. R. Langley, and J. P. Lefebvre. 2017. *Workbench Analysis Sequence Processor.* ORNL/TM-2017/698. Oak Ridge, Tennessee: Oak Ridge National Laboratory.

[13] Language Server Protocol Specification, [online] Available: https://microsoft.github.io/language-server-protocol/specification.

[14] Lefebvre, R. A., J. M. Scaglione, J. L. Peterson, P. Miller, G. Radulescu, K. Banerjee, K. R. Robb, A. B. Thompson, and J. P. Lefebvre. 2017. "Development of streamlined nuclear safety analysis tool for spent nuclear fuel applications." *Nucl. Technol.* 199(3).

[15] Hu, Rui. 2017. *SAM Theory Manual.* ANL/NE-17/4. Argonne, Illinois: Argonne National Laboratory. doi:10.2172/1353375.

[16] Stauff, N., N. Gaughan, and T. Kim. 2017. *ARC Integration into the NEAMS Workbench.* ANL/NE-17/31, September. Argonne, Illinois: Argonne National Laboratory.

[17] Stauff, N. E., R A. Lefebvre, L. Swiler, and B. T. Rearden, "Coupling of DAKOTA with the ARC suite of codes in the NEAMS Workbench for Uncertainty Quantification," ANS Summer meeting, Philadelphia, PA, USA, June 17–21, 2018

[18] Dominesey, K. A., M. D. Eklund, P. J. Kowal, and W. Ji, "Preliminary Integration of MCNP6 and PROTEUS into the NEAMS Workbench," ANS Summer meeting, Philadelphia, PA, USA, June 17–21, 2018.

[19] Bettini, L., "Implementing Domain-Specific Languages with Xtext and Xtend," 2nd ed., August 2016.

[20] Delchini, M. O., B. R. Langley, R. A. Lefebvre, W. D. Pointer, and R. T. Rearden. 2018. *Integration of the Nek5000 Computational Fluid Dynamics Code to the NEAMS Workbench.* ORNL/TM-2018/961. Oak Ridge, Tennessee: Oak Ridge National Laboratory.

[21] Salko, R., et al., "CTF Theory Manual," Oak Ridge National Laboratory, 2019.

[22] Salko, R., CTF Subkit, available online at https://code.ornl.gov/CTF/SubKit.

[23] Aysenur, T., R. Salko, M. N. Avramova, K. T. Clarno, and D. J. Kropaczek, "A New Fuel Modeling Capability, CTFFuel, with a Case Study on the Fuel Thermal Conductivity Degradation." United States. doi:10.1016/j.nucengdes.2018.11.010.