INL/EXT-21-64451

Light Water Reactor Sustainability Program

Technical Maturity Assessment of MOOSE-Based Tools: MASTODON, BISON, Grizzly

September 2021

U.S. Department of Energy

Office of Nuclear Energy

INL/EXT-21-64451

Technical Maturity Assessment of MOOSE-Based Tools: MASTODON, BISON and Grizzly

Yong-Joon Choi

September 2021

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy

EXECUTIVE SUMMARY

The United States nuclear industry is facing a strong challenge to maintain regulatory required levels of safety while ensuring economic competitiveness to stay in business. Safety remains a key parameter for all aspects related to operation of light water reactor (LWR) nuclear power plants (NPPs) and can be achieved more economically by using a risk-informed ecosystem such as that being developed by the Risk-Informed Systems Analysis (RISA) Pathway under the United States (U.S.) Department of Energy (DOE) Light Water Reactor Sustainability (LWRS) Program. The LWRS Program is promoting a wide range of research and development (R&D) activities with the goal to maximize both the safety and economically efficient performance of NPPs through improved scientific understanding, especially given that many plants are considering second license renewal.

The RISA Pathway has two main goals: 1) deployment of methodologies and technologies that enable better representation of safety margins and the factors that contribute to cost and safety, and 2) development of advanced applications that enable cost-effective plant operation.

The tools and methods used in the RISA Pathway should have high confidence and highest technical maturity for implementation to industry. They should also have a capability to support risk-informed decision-making for both probabilistic and deterministic elements of safety. Therefore, the RISA Pathway performs a comprehensive assessment of verification and validation (V&V) status of various software tools to enhance their credibility and expedite deployment to industry.

This report summarizes results of the technical maturity assessment of the computational fluid dynamics (CFD) and high fidelity simulation tools MASTODON, BISON, and Grizzly. All these tools are considered a part of the MOOSE-based toolkit. The report includes overview of V&V status, specific details such as tool capability and features, quality assurance program, developer/independent V&V records, separation/integral tests history, and software documentation (e.g., user's manual).

CONTENTS

FIGURES

TABLES

ACRONYMS

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
AOO	Anticipated Operation Occurrence
ASME	American Society of Mechanical Engineering
ASR	alkali-silica reaction
ATF	accident tolerant fuel
BEPU	best-estimate plus uncertainty
BWR	Boiling Water Reactor
CDF	cumulative distribution function; core damage frequency
CFD	computational fluid dynamics
CSAU	code scaling, applicability, and uncertainty
DBA	Design Basis Accident
DNC	Dynamic Natural Convection
DOE	U.S. Department of Energy
EPRI	Electric Power Research Institute
ER	equipment reliability
FOM	figure of merit
FY	fiscal year
GUI	graphic user interface
IEEE	Institute of Electrical and Electronics Engineers
INL	Idaho National Laboratory
LOCA	Loss of Coolant Accident
LWR	Light Water Reactor
LWRS	Light Water Reactor Sustainability
MAAP	Modular Accident Analysis Program
MOOSE	Multiphysics Object-Oriented Simulation Environment
MP-BEPU	Multi-Physics Best-Estimate Plus Uncertainty
NASA	National Aeronautics and Space Administration
NEI	Nuclear Energy Institute
NEUP	Nuclear Energy University Program
NPP	nuclear power plant
NQA	nuclear quality assurance

NRC	U.S. Nuclear Regulatory Commission
PDE	partial differential equation
PDF	probability density function
PIRT	phenomena identification and ranking technique
PRA	Probabilistic Risk Assessment
PWR	pressurized water reactor
QA	quality assurance
QAP	Quality Assurance Program
R&D	research and development
RAVEN	risk analysis in a virtual environment
RISA	Risk-Informed Systems Analysis
RIA	Reactivity Initiated Accident
RIMM	Risk-Informed Margin Management
RI-MP-BEPU	Risk-Informed Multi-Physics Best-Estimate Plus Uncertainty
RISA	Risk-Informed Systems Analysis
RISMC	Risk-Informed Safety Margin Categorization
RTM	Requirement Traceability Matrix
SLR	second license renewal
SQA	software quality assurance
SQAP	Software Quality Assurance Plan
SQE	software quality engineering
SRQ	System Response Quantify
SSC	system, structure, and component
TMP	Technology Maturation Plan
TRA	technology readiness assessments
TRL	Technology Readiness Level
V&V	verification and validation

1. INTRODUCTION

1.1 Background

The risk-informed tools and methods, so-called RISA Toolkit, to be used in industry needs an appropriate level of verification, validation, and uncertainty characterization to ensure maximum credibility for safe nuclear power plant operation. A unique focus of our assessment is on the maturity and usability of a specific tool. Since the tools and methods used in the RISA Pathway need to have highest technical maturity and could be used by industry immediately, the comprehensive assessment of verification and validation (V&V) status an important task for the toolkit successful industry deployment [1].

Most of tools have developer and/or user V&V program to guarantee quality. However, V&V programs still require an assessment to verify their technical maturity based on the philosophy of risk-informed applications, i.e., tools readiness to be used in conjunction with the Probabilistic Risk Assessment (PRA). Therefore, the RISA Pathway uses the following goals to assure RISA Toolkit quality:

- Define requirements based on risk-informed concept
- Investigate and review development and V&V status for technical maturity assessment
- Identify technical gaps and propose improvements to meet RISA Toolkit requirements

The first part of the work was to collect and summarizes available information of selected RISA Toolkit. This includes list of documents and accessibility of the V&V records for each version of the software, if necessary. The selected toolkit was then evaluated for its maturity level. Based on the risk-informed concept, the capability and/or applicability of PRA method is the main requirement for the toolkit being assessed. The high level requirements were set based on the Requirement Traceability Matrix (RTM) to capture the requirements from users and developers of the project or software. The importance of each requirement was evaluated by the Phenomena Identification and Ranking Technology (PIRT) method which gives a systematic way of gathering information from experts on a specific subject and ranking the importance of the information in order to determine the highest priority for the research. Finally, Technology Readiness Level (TRL) was used to measure the level of maturity. Developed by the National Aeronautics and Space Administration (NASA), the TRL is a method for estimating the maturity of technologies during their development and acquisition phases. A total of nine levels are set from low level (1~4) to high (5~9), which represent phases from the initial research to ready for industrial use. Figure 1-1 shows an example of TRL increase.

The industry application pilot demonstration projects (pilot project in-short) are main features of the RISA Pathway. The pilot projects focus on specific scope of phenomena, components, and simulation capabilities needed to address a given area. As a part of these projects, refinement of the associated methods and tools continues at a reduced level of effort compared to the effort associated with the tool development. However, not all tools are suitable for use in RISA pilot projects, thus, certain tools may need additional development. The RISA Pathway identifies technical gaps of a target toolkit and proposes additional development to meet requirements of risk-informed applications.

Figure 1-1: Schematic diagram of TRL improvement in RISA Pathway

As the development and capabilities of the RISA Toolkit progresses, the pathway will collaborate with industry to determine how to transition the RISA Toolkit to a user-supported community of practice. The assessment of V&V status project supports a smooth industry deployment of RISA Toolkit including planning for lifecycle software management issues such as training, software quality assurance, and development support.

In FY-2019, technical maturity was assessed for RISA thermal-hydraulics tool RELAP5-3D and multi-purpose PRA tool RAVEN [2]. Both of tools were found to have a sufficient level of technical maturity and proposed to be used in the RISA Pathway and deployed to industry. The outcome of this report supported developers and users to understand the current level of technical maturity and clarified the need of additional development to address technical gaps before tool's deployment. As the result, the best-estimate plus uncertainties (BEPU) capabilities are under development for RELAP5-3D.

In FY-2020, the risk assessment tool EMRALD and human reliability assessment tool HUNTER were reviewed [3]. Both of the tools are still under development and their technical maturity levels were evaluated as low. Especially, HUNTER was not formed as computer software, and a new project was proposed in FY-2021.

1.2 List of RISA Toolkit

Based on the current and potential RISA pilot projects, Figure 1-2 shows the list of computational software proposed to be used and could be deployed to industry for risk-informed margin management. However, the list is not all-inclusive. Many of the listed tools are completely developed and currently used in various industries. However, some tools are still under development and may need additional V&V activities.

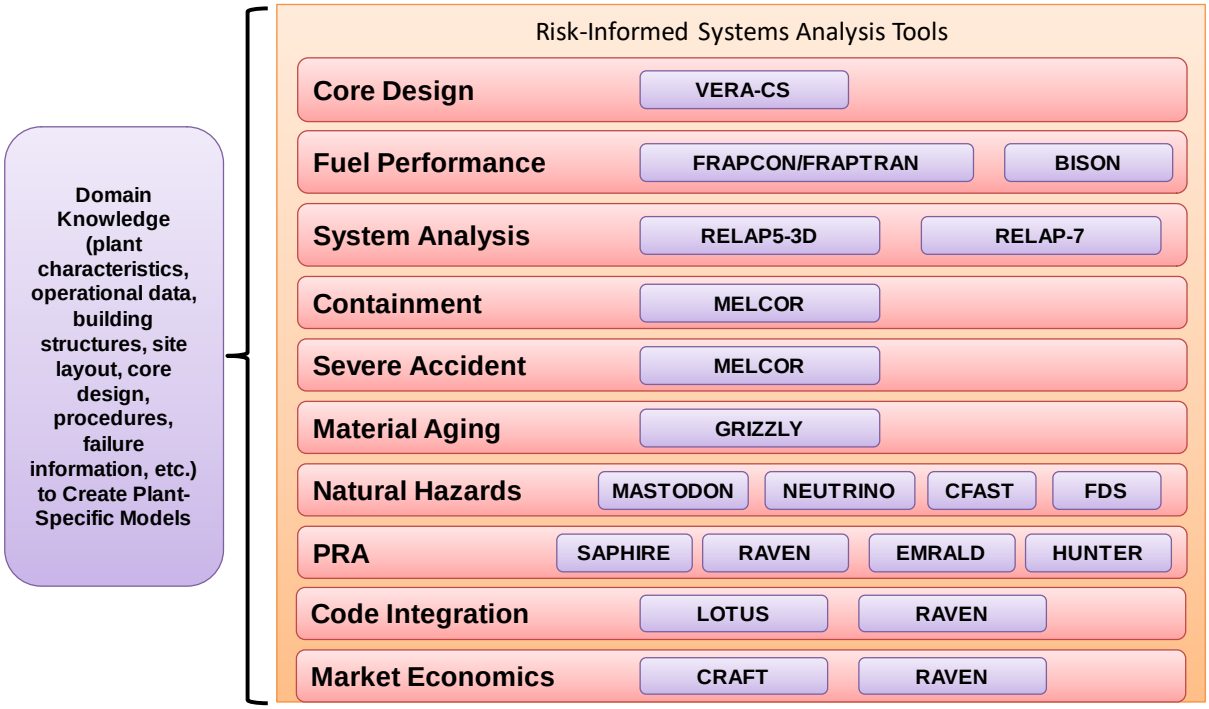

Figure 1-2: Current software tools used to perform risk-informed analyses.

1.3 RISA Toolkit Deployment Plan

The RISA Toolkits are sets of computer software that is used in the industry application pilot demonstrations and are planned to be deployed to the U.S. nuclear industry to support risk-informed

margin management (RIMM) analyses. The RISA Pathway reviews software V&V to provide a clear understanding of the software V&V status to the future RISA Toolkits users The RISA Pathway will continue to communicate with industry to identify issues and collect feedback. The RISA Toolkit deployment has the following four-step process:

1. Select tools and methods.
2. Confirm verification and validation status.
3. Pilot demonstrations using selected tools and methods.
4. Industry deployment and feedback.

In order to meet the above four steps for a tool development, the RISA Pathway proposes a five-year project plan as shown in Figure 1-3. For the first year, the RISA Pathway focuses on the initiation of selected pilot demonstration projects and delivers preliminary results on the case studies. The feedback on selected pilot demonstration projects is solicited from industry through the RISA Pathway working team. Based on the preliminary studies, the pilot demonstration projects are extended to a full-scale R&D during the next two years. The validation and verification of associated the RISA Toolkit is done during this period. For the last two years of the project, the RISA Pathway develops optimized methods of R&D, advances the toolkit to implementation to industry, and develops a long-term support plan. It is noted that the project timeline could be different depending on the toolkit complexity.

1.3.1 Selection of RISA Toolkit

The tools and methods used in the RISA Pathway should have high confidence and appropriate technical maturity and ability to cover a wide RIMM area range. Advanced technologies should be applied to the RISA Toolkits, such as multi-physics and multi-scale analysis, and cutting-edge computational proficiency as well as the capability of uncertainty control if necessary. The toolkit should also have the capability to support risk-informed decision-making for both probabilistic and deterministic elements of safety. The current RISA Toolkits include various computer simulation tools that can cover a wide range of work scopes. Many of tools are currently available and used by industry, and they are well validated with mature technology level. However, other tools are still under development and need to be verified and/or validated to confirm their technical maturity and suitability for the RISA Pathway framework.

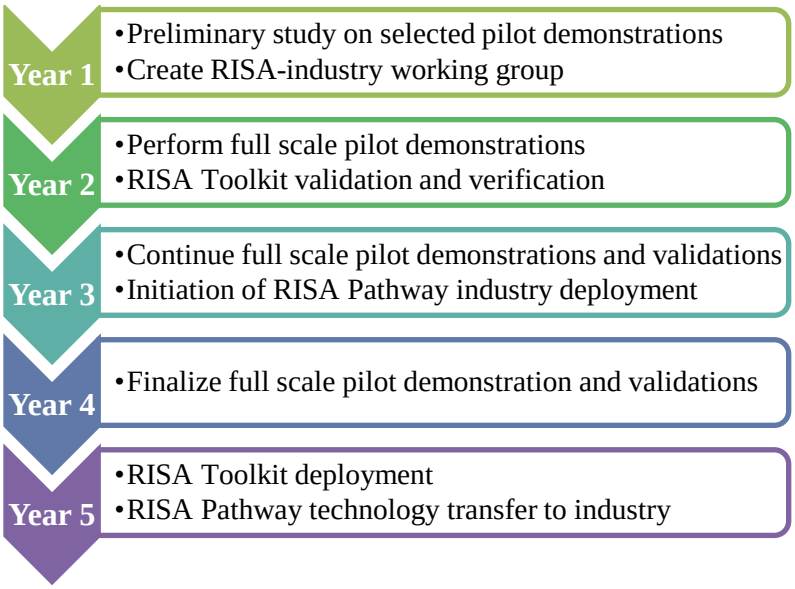

Figure 1-3: Notional 5-year plan of a RISA Pathway industry application

1.3.2 Verification and Validation of the RISA Toolkit

To provide confidence during industry deployment, the selected tools and methods should address quality assurance levels appropriate for industry use. Well-known methods such as V&V and uncertainty quantification of the produced result will enhance credibility of the selected tools and enable industry to use them with confidence. The selected RISA Toolkits are examined to confirm V&V status to show their TRL and to assure quality of the outcomes. The RISA Pathway delivers annual reports of V&V examination and confirmation for the selected RISA Toolkits. The reports include specific information of the tool, such as capability and features, quality assurance program, developer/independent V&V record, separation/integral tests history, user documents, and feedback.

2. METHODS FOR TECHNICAL MATURITY EVALUATION

A given RISA Pathway toolkit represents related computer software to be used for pilot demonstration projects as discussed in Section 1.2. Since simulation appears to be the only feasible option to capture realistic aspects of the multi-physics behavior of a complex system, the software user should clearly understand features, characteristics, and any potential issues of the selected tool. The RISA Pathway focuses on the use of modeling and simulation tools for risk-informed approach. As such, software tools and methods must have capabilities to support risk-informed analyses (e.g., use of probabilistic inputs instead of deterministic, specific incorporation of uncertainties). This implies that the software should have its own PRA capability, or it could be coupled with other PRA tools.

The technical maturity evaluation of the RISA Toolkit focuses on the capability, usability, and applicability to risk-informed applications. If the toolkit is still under development, the developer must have responsibility for further maintenance once development is finalized. Version control, V&V history, license maintenance, quality assurance programs, and customer service are also required for the sustainable use of the software. The RISA Toolkit assessment reports summarize the above information to facilitate deployment of the tools to industry.

Figure 2-1 is the schematic diagram of RISA Toolkit maturity evaluation process. The list of requirements that toolkit should fulfill is selected from the RTM method. The RTM is then evaluated by PIRT to categorize importance of the requirements. Three levels of ranking, high - medium - low, are used for each requirement. Finally, the TRL method provides a maturity level of each requirement.

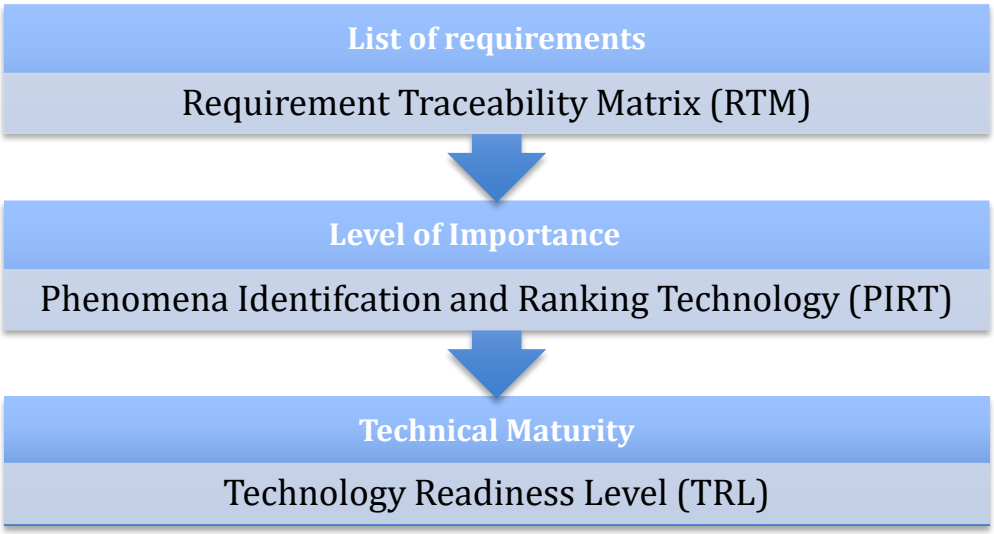

Figure 2-1: RISA Toolkit maturity evaluation procedure

2.1 Requirement Traceability Matrix

The RTM is created to associate specific requirements with the work products that satisfy them. Tests are also associated with the requirements on which they are based so a documented proof exists that the product has met the requirement. The matrix provides unique identifiers for each requirement and ensures completeness that lower-level requirements come from higher-level requirements. This matrix provides a tangible item that can be examined by the customer and the developer alike. Information that may not be clearly explained in descriptive text is directly traceable forwards and backwards from a requirement or from an associated test. Traceability is used to manage changes and provides a basis for test planning. It is a tool to ensure that the software process has been completed from the initial definition to completion of a product. The use of RTM provides a software quality checkpoint. It is however noted that the RISA

Toolkit V&V status assessment does not use a traceability matrix but specifies requirements for the target software.

The requirements for RISA Toolkit focus on the higher technical maturity and capability or applicability of risk-informed methods. The following requirements are used for the RISA Toolkits.

Development Level
RISA Toolkit should have the highest development level to promote an immediate use by industry. If the tool is still under development, the goal should clearly indicate the commercialization plan. The use of an under-developed or lower technical maturity tool will decrease credibility of risk-informed analysis outcomes.

Use of Proven Technology
Some of newly developed tools use cutting-edge technology to build the software. However, such forefront technology may not always be applicable to existing approaches. The candidate of RISA Toolkit should be applicable to the risk-informed applications for LWRs.

PRA Capability/Applicability
Since the main goal of a risk-informed analysis is to support safety margin management quantifies using PRA, the RISA Toolkit should have its own PRA capability or applicability to an existing PRA tool.

Documentation
RISA Toolkit should have a recognizable list of documents which user can access easily, including user manual, theory manual, development plan, quality assurance (QA) plan, V&V plan and results, etc.

System Requirements
Users may use diverse computer and operating systems, and RISA Toolkit should be operable in different computer systems. Hence, the system requirements and related verification test result should be properly documented.

Easy Installation
RISA Toolkit should be installed easily without developer's support. Facility of installation is be reviewed.

Graphic user Interface (GUI)
GUI will provide applicable information to users. Any type of graphical output generator will be listed.

Version Control
For every step of a software update, the version history of the tool should be maintained. Related documents and manuals should also be updated.

V&V History
In high-level definition, verification is to check that the tool is correctly built while validation is a confirmation that the outcome of the verified code is correct. Since V&V result is one of most important parameters of the quality of the code, the history of V&V activities by developers and/or third party should be documented. This includes regression test list for code verification.

QA Program
Quality assurance program ensures the quality of the RISA Toolkit. It includes the ASME Nuclear Quality Assurance (NQA-1) certification program to support nuclear industry.

Webpage
A dedicated webpage should be maintained in order to communicate with users.

User Support
The code developer should provide feedback on the problems reported from users.

Training Program

For easy deployment to industry, a training program by the developer or RISA Pathway is a good solution.

License
The developer should clearly indicate details of licensing.

2.2 Phenomena Identification and Ranking Technique

The PIRT is a method to systematically gather information from experts on a specific issue (e.g., NPP accident) and rank the importance of the information. The goal is to support decision-making such as determining which information should have high priority for research on that subject. The PIRT was first developed in the late 1980s and has been successfully applied in the nuclear industry such as nuclear reactor safety analysis. In general, experts determine the relative importance of phenomena by considering their influence on the relevant plant safety metrics (e.g., peak cladding temperature).

For the RISA Toolkit V&V status assessment, three levels are applied to the requirements set in Section 2.1 by RTM:

High: Dominate impactful requirement. Require high level of technical maturity.

Medium: Moderate impactful requirement. Require medium level of technical maturity.

Low: Small or not impactful requirement. Satisfy with basic level of technical maturity.

2.3 Technology Readiness Level

TRL is a type of measurement system used to assess the maturity level of a particular technology. Each technology project is evaluated against the parameters for each technology level and is then assigned a TRL rating based on the projects progress. There are nine technology readiness levels as shown in Figure 2-2. TRL 1 is the lowest and TRL 9 is the highest. When a technology is at TRL 1, scientific research is at the beginning and those results are being translated into future research and development. TRL 2 occurs once the basic principles have been studied and practical applications can be applied to those initial findings. TRL 2 technology is very speculative, as there is little to no experimental proof-of-concept for the technology. When active research and design begin, a technology is elevated to TRL 3. Generally, both analytical and laboratory studies are required at this level to see if a technology is viable and ready to proceed further through the development process. Often during TRL 3, a proof-of-concept model is constructed.

Once the proof-of-concept technology is ready, the technology advances to TRL 4. During TRL 4, multiple component pieces are tested with one another. TRL 5 is a continuation of TRL 4, however, a technology that is at TRL 5 is identified as a breadboard technology and must undergo more rigorous testing than technology that is only at TRL 4. Simulations should be run in environments that are as close to realistic as possible. Once the testing of TRL 5 is complete, a technology may advance to TRL 6. A TRL 6 technology has a fully functional prototype or representational model. TRL 7 technology requires that the working model or prototype be demonstrated in a space environment. TRL 8 technology has been tested and "flight qualified" and it's ready for implementation into an already existing technology or technology system. Once a technology has been "flight proven" during a successful mission, it can be called TRL 9. The U.S. Department of Energy proposed Technology Readiness Assessments (TRAs) and developing Technology Maturation Plans (TMPs) to assist individuals and teams that will be involved in conducting program and project management for the acquisition of capital assets [7]. TRAs and TMPs activities assist in identifying technology risks and enable the correct quantification of scope, cost and schedule impacts in the project. Recent application of TRL in nuclear industry was to review maturity of advanced nuclear fuels and material development [8]. The purpose of the research was to evaluate existing technologies and identify R&D needs while developing advanced fuels and materials. Similar to the RTM list, required parameters for critical issues are defined and measured.

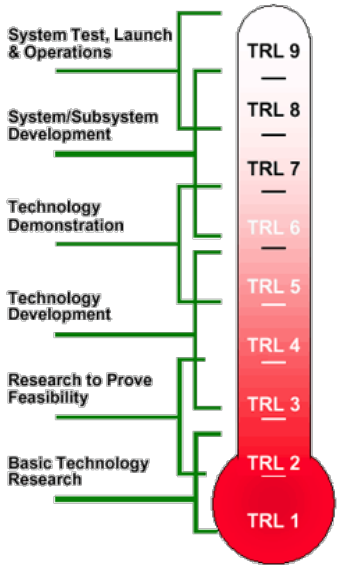

Figure 2-2: Schematic diagram of Technology Readiness Level

Based on general TRL, specific TRL for RISA Toolkit is proposed in Table 2-1. The RISA Pathway aims deploy higher TRL toolkit to industry, which is TRL 7 or higher. For the RISA Toolkit, the industry pilot demonstration project is the best practice to upgrade from TRL 6 or lower to TRL 7 or higher.

Table 2-1: Description of Technology Readiness Level for RISA Toolkit

Level of Tech. Development	Technology Readiness Level	TRL Definition	Application to RISA Toolkit
System Operation	TRL 9	Actual system operated over the full range of expected conditions	The toolkit technology is in its final form and routinely used under the full range of industrial purpose.
System Development	TRL 8	Actual system completed and qualified through test and demonstration	The toolkit has been proven to operate in its final form and under expected conditions. In almost all cases, this TRL represents the complete of R&D. Entire planned and proposed V&V of the toolkit are finalized by developer/user.
	TRL 7	Full-scale, similar (prototypical) system demonstrated in relevant environment	Full-scale demonstration of actual/prototypical technology/toolkit in relevant operation environment. Full-scale experiment and validation are performed. Development of the toolkit is virtually complete.
Technology Demonstration	TRL 6	Engineering/pilot-scale, similar (prototypical) system validation in relevant environment	Engineering-scale models or prototypes are tested in a relevant environment. Experiment and validation in engineering/pilot-scale environment including scaling effect testing, which can support operational system, design. This level represents completion of technology development for operational demonstration and prototype toolkit is ready for test. The prototype toolkit will have capability of performing all the functions that will be required from actual operational system. The operating environment for the testing should closely represent the actual operating environment. Major difference between TRL 5 is the scale-up from laboratory to engineering size.
Technology Development	TRL5	Laboratory scale, similar system validation in relevant environment	The basis of technology is fully integrated into the toolkit and ready for larger scale demonstration and validation. This level will include results from the laboratory scale test, analysis of the differences between the laboratory and actual operating system/environment, and analysis of experiments/demonstrations results for actual operating system/environment application. The major difference between TRL 4 and 5 is the increase of the toolkit fidelity and environment to

Level of Tech. Development	Technology Readiness Level	TRL Definition	Application to RISA Toolkit
			the actual application. Verification is complete and the toolkit development level is close to prototypical.
	TRL 4	Component and/or system validation in laboratory environment	The basis of technology for toolkit is partly integrated and can be apply for component level demonstration. This is relatively "low fidelity" compared with the actual level of toolkit completion level. The expected maturity of this level includes the integrated experiments and validation, examination of scaling effect and actual application. Verification and regression test could be included. TRL 4-6 represents the bridge from scientific research to engineering. TRL 4 is the first step in determining whether the basic modeling will work in the toolkit.
Research to Prove Feasibility	TRL 3	Analytical and experimental critical function and/or characteristic proof-o-concept	Actual R&D is started for the toolkit development. This includes analytical studies and laboratory scale studies to validate the phenomena of separate technology. This level will have results of laboratory tests performed to measure parameters of interest and comparison to analytical predictions for critical toolkit functions. At TRL 3, actual R&D progresses to experiments and verifications. Validation could be done for part of the toolkit development, but system-level validation is not yet initiated.
Basic Technology Research	TRL 2	Technology concept and/or application formulated	Progressed from TRL 1, technical options may be developed in TRL 2. However, still no activity was performed to prove assumptions and concept. Literature studies outline the toolkit development concept. Most of activity in this level is analytical research and paper studies to understand the goal of the R&D. Related experiments and V&V works could be designed during this level.
	TRL 1	Basic principles observed and reported	This is the lowest level of technology readiness. Scientific research begins to be translated into applied R&D. Available information includes published research or other references that identify the principles that underline the technology. No actual R&D started.

3. GUIDANCE FOR SOFTWARE V&V AND QA

This chapter summarizes the guidance for software V&V and QA to give insights for managing quality of the RISA Toolkit during its development and industry deployment. It is recommended that developers should use this guidance to improve credibility and quality of the software being developed.

3.1 V&V of Computational Fluid Dynamics Tools

The computational fluid dynamics (CFD) tools have numerical methodologies with high fidelity, and the approach for technical maturity assessment could be different compared to low fidelity tools. The verification strategy of the CFD tools is to identify and quantify errors from the computational model and its solution compared to analytical and highly accurate numerical solutions. This can be accomplished by comparison of computation results with experimental data by quantifying both errors and uncertainties. The code-to-code and code-to-experiment benchmarks could be also used for V&V of the CFD tools. For the uncertainty analysis of the tools, both aleatory (i.e., due to randomness) and epistemic (i.e., due to lack of knowledge or data) uncertainties need to be assessed to improve code credibility and reliability.

3.1.1 Definition of V&V for CFD

In general, V&V aims to provide accuracy and improve reliability of computational tools and its results [6]. The IEEE (Institute of Electrical and Electronics Engineers) definitions of V&V are:

- Verification: the process of evaluating the products of a software development phase to provide assurance that they meet the requirements defined for them by the previous phase
- Validation: the process of testing a computer program and evaluating the results to ensure compliance with specific requirements

In practical use, the verification means that whether the conceptual model which user wants to use is correctly solved through the numerical mathematics in the computer code. The verification does not consider the relationship between conceptual model and actual physics. The validation focuses on the accuracy of the computational model to simulate actual physics assuming that the verification of the tool is completed. In this sense, the verification could be considered as the first stage of the validation. The overall V&V process should address and account for uncertainties throughout the problem, including material property uncertainties and experimental or field data (e.g., boundary conditions and instrumentation uncertainties).

3.1.2 Graphical Comparison

One of easiest way of V&V is the graphical comparison of the results. The V&V of the CFD tools mainly started in the field of aerodynamics. Similar to V&V of the low fidelity tools, graphical comparison between the simulation results and experimental results is one of easiest approach, but it has limitation in quantifying numerical errors and uncertainties. In addition, this method is not suitable for taking into account initial and boundary conditions. However, graphical comparison could provide a general guidance on “acceptability” of the result as a basic V&V method. In order to give credibility to the graphical comparison method, quantification could be performed by evaluation of variation of appropriate metrics with respect to experimental results, which would also provide quantified results for both errors and uncertainties of the model.

3.1.3 Verification Process

The general process of the CFD tool verification is determination of the model accuracy. This includes identification, quantification and reduction of the errors from the computational model and solutions. To do this, a benchmark study is necessary to compare with highly accurate and reliable result, preferably experimental results. In general, the code developer performs verification of the code during

the early stage of the code development. However, verification needs to be repeated once the code has been subsequently modified or enhanced.

Figure 3-1 describes the verification process of comparing computational solutions from the code with various highly accurate solutions. Once the developer set conceptual model for the target simulation, the computational model provides computational solution. The verification test is to compare the result between the computational solution and highly accurate solutions. The highly accurate solutions refer to either analytical solutions or other highly accurate numerical solutions. However, highly accurate solutions are only available within simplified numerical model, which may not be available for high fidelity CFD tools. Hence, verification of CFD needs to provide evidence that the discrete numerical models in the CFD tool correctly solve the target conceptual model, thus the accuracy requirement of the tool needs to be fulfilled. It is noted that the meaning of "correct solving" is the area of mathematics and computer science, not physically correct. Validation is a physical science and mathematics.

3.1.3.1 Quantification of Error

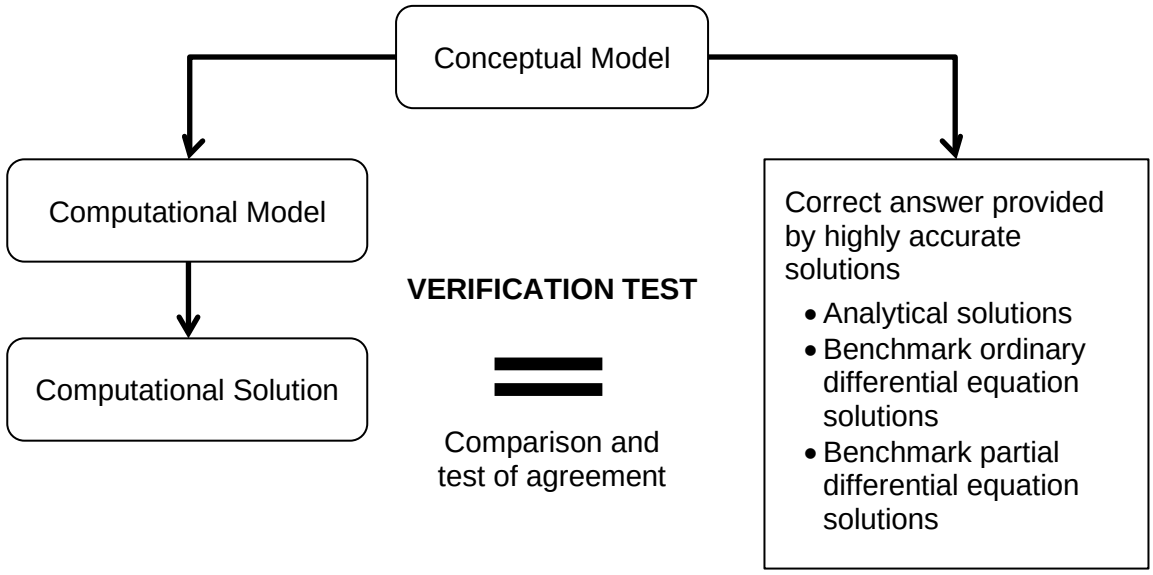

Figure 3-1: Schematic diagram of verification process [6]

Source of error

The emphasis in verification is the identification and quantification of error, as well as the demonstration of the stability, consistency, and robustness of the numerical scheme. Assuming the numerical procedure is stable, consistent and robust, following are the main source of errors could be occurred during the computational methods:

- Insufficient spatial discretization convergence
- Insufficient time discretization convergence
- Insufficient iterative procedure convergence
- Computer round-off
- Computer programming

While the analytical or formal error analysis is unnecessary for the verification process, demonstration of the error bounds is required, which could be quantified from global or local error norm. The error bounds could be used for the test of agreement assessment shown in Figure 3-1.

The first four error sources listed above could be referred to as solution verification or solution accuracy assessment. The fifth error source, the programming errors can occur in input data files, source code programming, output data files, compilers, and operating systems, are addressed using methods and tools in software quality assurance (SQA) and engineering (SQE). The identification of programming errors could be referred to as code verification, as opposed to solution verification.

Error, E, in a CFD calculation can be defined as

$$E = S_{exact} - S_{discrete} \qquad \text{Eq. (1)}$$

Here, S_{exact} is the mathematically correct solution of the conceptual model and $S_{discrete}$ is numerical solution from given discrete approximation of the computational model. Both solutions have same initial and boundary conditions. Error can be characterized by space (h) and time (τ) discretization, and from triangle inequality, estimated norm value of E can be written as

$$E \leq \| S_{exact} - S_{h,\tau \to 0} \| + \| S_{h,\tau \to 0} - S_{h,\tau,I,c} \| < \varepsilon \qquad \text{Eq. (2)}$$

where, $S_{h,\tau,I,c}$ is the discrete solution in given space (h), time (τ), convergence parameter (I), and computer (c). ε is an arbitrary small number as accuracy bound. $S_{h,\tau \to 0}$ is the limit of $h \to 0$ and $\tau \to 0$ of the exact solution. The first term of Eq. (2), $\| S_{exact} - S_{h,\tau \to 0} \|$, is the error generated by the exact solution of numerical discrete equations. This value will converge to zero if the numerical scheme is stable or problem has very simple space and time discretization. However, the second tem, $\| S_{h,\tau \to 0} - S_{h,\tau,I,c} \|$, is always non-zero value because $S_{h,\tau,I,c}$ term has finite values. This term also includes errors from computational system, source code and programming, and can be very large in case of complex problems. The size of $S_{h,\tau,I,c}$ term could be minimized by reducing number of spatial discretization (h), in other words error from 1-D problem is always smaller than 2 or 3-D problems. Hence, the verification activity need to focus on quantifying the value of $\| S_{h,\tau \to 0} - S_{h,\tau,I,c} \|$ term in Eq. (2).

Error Estimation in Verification

The discrete approximation computational model (i.e., partial differential equation: PDE) introduces two types of numerical errors: truncation and discretization errors. The truncation error is the difference between true or analytical derivative of the equation and its derivatives obtained by the numerical approximation, thus, it is the unavoidable error caused during approximating of the mathematical processes such as finite difference or finite volume approximations. The discretization error, however, is the error introduced during discretization of PDE of target physics, which means the error resulted by modeling of continuous variable of physics to the finite number of numerical spaces. If the discretization method is “strongly consistent” and the truncation error converges to zero, then the discretization error could be minimized or eliminated by using very fine spatial discretization. In other words, the discretization error is proportional to truncation error. Correct understanding of both truncation and discretization errors of computational model used CFD is one of most important processes for quantification of errors during the verification stage.

The truncation error, TE, could be expressed as $TE(h^p, \tau^q)$ with $p>0$ and $q>0$. Then when discrete solution is converged, the discretization error by using first term of Eq. (2) is

$$\| S_{exact} - S_{h,\tau} \| = TE(h^p, \tau^q) \qquad \text{Eq. (3)}$$

Here, the norm gives an appropriate measure of the difference between solutions of the discretization and exact equations. For non-uniform gird problem, Eq. (3) becomes local estimation of the discrete error over the domain of the discretization.

For the verification process, the discretization error could be used to answer following questions:

- Does numerical solution converge to exact solution when mesh size convergences to zero?
- What is the effective order of (*p* and *q* in Eq. (3)) error actually appeared in the calculations?
- What is the discretization error observed during actual simulation?

Two approaches are available to answer above questions: a priori information and a posteriori information methods. The term a priori method uses evaluation of error from computational model by using the available information from the partial differential operators, numerical algorithms and associated initial and boundary data [7]. On the other hand, a posteriori method also considers solutions from the numerical models to estimate errors [8]. Hence, the a posteriori method generally becomes "empirical" approach, which is more practical in CFD verification. For example, the errors of the verification test with systematic refinement of mesh size and time step will approach to zero if numerical model is robust and discretization method is "strongly consistent". If the order of the accuracy is also shown constant as the mesh and time step are reduced, it could be defined that the errors are in "asymptotic region" of the discretization. The empirical approach in the asymptotic region is, therefore, a key to solve the above questions during the verification. The empirical approach demands large amount of computational resources, and usually applied on simplified model problems or concentrate on observing little change in important variables during mesh size and time step convergence studies.

As an application of a posteriori error estimation, the true confidence of CFD calculation could be achieved by both time and space convergence within sufficient discretization resolution. The computation error can be then expressed as

$$\| S^{r1} - S^{r2} \| < \varepsilon \qquad \text{Eq. (4)}$$

where, *r1* is first level of time and space refinement of discretization, *r2* is more refined level than *r1*, and ε is the accuracy required in the calculation. To find exact value of error in Eq. (4), the Richardson extrapolation method can be used. For both *r1* and *r2*, the exact values are

$$S_{exact} = S_{h1} + \alpha h_1^p + TE(h_1^{p+1}),$$

$$S_{exact} = S_{h2} + \alpha h_2^p + TE(h_2^{p+1}) \qquad \text{Eq. (5)}$$

where, α is constant, with assumption of $h_1 > h_2$. Arranging two equations into one by removing α, then, the exact value becomes

$$S_{exact} = \left[\frac{h_2^p S_{h1} - h_1^p S_{h2}}{h_2^p - h_1^p}\right] + TE\left(h_1^{p+1}\right) + TE\left(h_2^{p+1}\right) \qquad \text{Eq. (6)}$$

The term in brackets of Eq. (6) represents an extrapolation of discrete solution toward the exact solution that is one-order accuracy higher than *p*. Therefore, the discretization error for given exact value S_{h1} is

$$\|S_{exact} - S_{h1}\| \cong \left|\left[\frac{S_{h2} - S_{h1}}{h_2^p - h_1^p}\right] h_1^p\right| + TE\left(h_1^{p+1}\right) + TE\left(h_2^{p+1}\right) \qquad \text{Eq. (7)}$$

Above Eq. (7) estimates a posteriori error, which can be used for accuracy bound analysis during CFD verification. Per Richardson extrapolation, the error term could be first and/or second order. Figure 3-2 is an example of spatial error estimation with different order error tem as mesh size (*h*) varies.

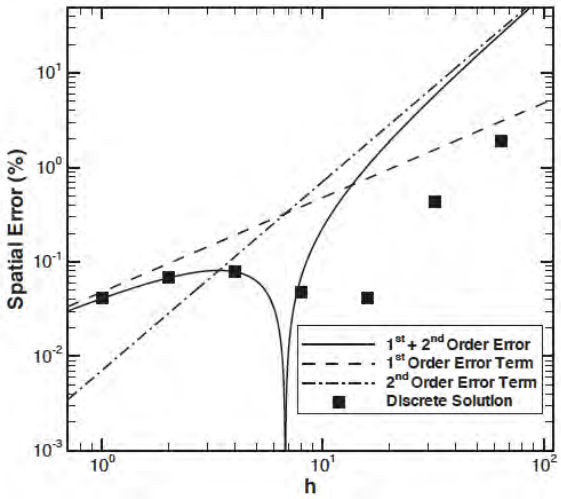

Figure 3-2: Example of extrapolated error estimation [25]

3.1.3.2 Verification Testing

Testing for the verification purpose is the critical area for CFD verification. One of popular approach is to use of "T test experiment" [10]. Two tests are proposed: T1, the static tests, and T2, the dynamic tests. The static test is the method to analyze the form, structure, and consistency of the code without executing the code. This includes software and manual reviews, inspections, audits, and data flow analyses. The dynamic test involves execution of the code, and the result of code execution is used to estimate code error or issues during code design. Figure 3-3 shows integration matrix of a set of activities in top-down verification process for CFD. As an appropriate approach to CFD verification activities, especially for high-consequence computing, it should include all of the indicated elements in code and calculation verification.

Code Verification

The process of software quality engineering at INL is explained in Section 3.3.

The code verification includes practices and standards associated with software quality engineering. The software quality testing is divided into two categories: static, and dynamic, and dynamic testing includes regression, black box and glass box testing. The software quality testing is the most important step for minimizing the number of software bugs that may be present in CFD. It is noted that algorithm testing is always necessary whether the software quality testing is performed or not.

Regression testing is one of most popular methods for code verification and for re-evaluation of the result accuracy. The definition of regression test is "any repetition of tests (usually after software or data change) intended to show that the software's behavior is unchanged except insofar as required by the change to the software or data" [6]. In practical use, regression testing is to confirm code features are performing as expected under the specific computer system. It is common to perform automatic regression testing during code installation to new system and code development and modification, and record its result as accuracy estimation of the code. In case of code modification, regression testing is to minimize efforts on fixing errors in the modified version if the tests are failed. Generally, regression testing needs a considerable amount of human and financial resources in performing the tests, evaluating the results and correction of errors, and maintaining the testing capability.

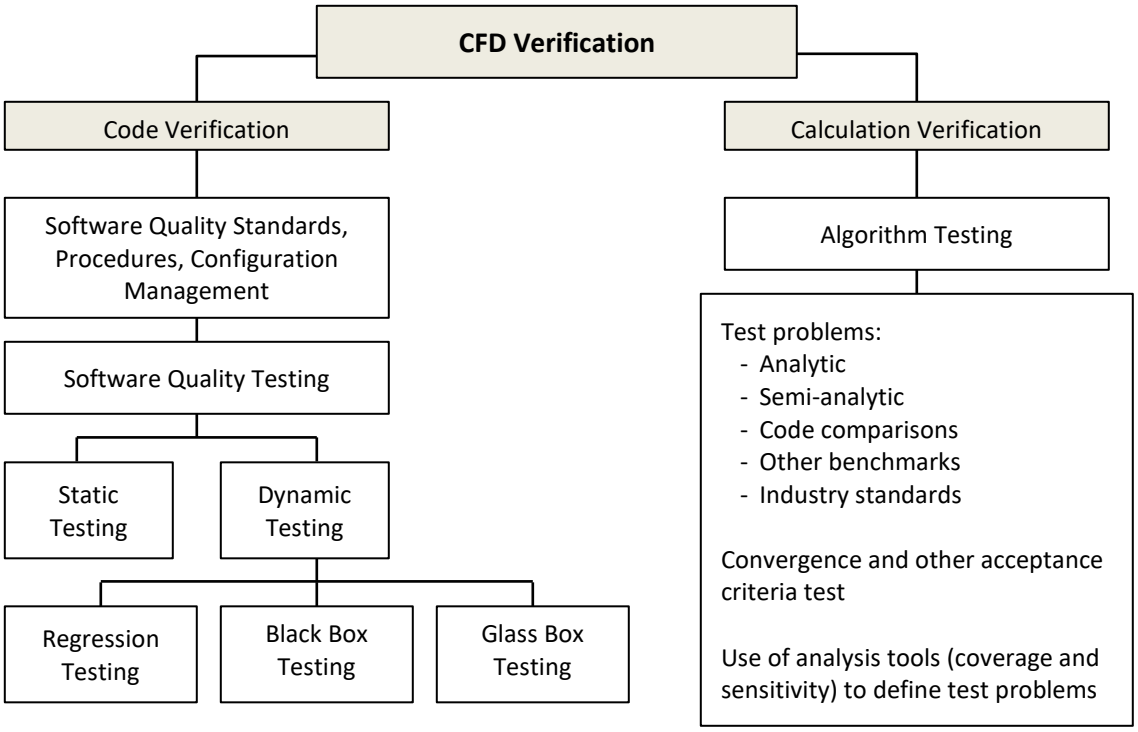

Figure 3-3: CFD verification matrix

The glass box testing means path testing for the code developers to understand code design and architecture is correctly formulated as desired based on the code development plan and is commonly used during SQE. The black box testing is functional testing by users who did not involve in code development or the developer who aims to investigate empirical performance of the code. Both glass box and black box tests could be used for regression test.

Calculation Verification

The calculation verification uses the algorithm testing with emphasizing the numerical correctness and performance of calculation algorithms. The major activity includes definition of appropriate test problems for assessing solution accuracy and assessment procedures for judging numerical performance versus these tests. For CFD algorithm testing the developer must set: goals and logic of the testing, method and principles of the testing, nature of comparison and acceptance criteria, and precise definitions of success or failure of the testing.

The logic of the testing must be clear. For example, CFD code can organize logical test for (1) test with exact analytical solutions, (2) test with semi-analytic solutions, and (3) benchmark (semi-) analytical solutions with PDE solutions.

Industry standards are also useful for algorithm testing. Many of scientific and engineering communities have been developing various types of testing problems on its own physics discipline with focus on code comparison standards and acceptance or success criteria. If the industrial standards are not available, test problems could be selected by consensus of the community associated with both developers and users.

Mesh and time step size convergence study is basic test cases of calculation verification. Developers and users can define their own parameters of interest to estimate accuracy of CFD code convergence and model sensitivity study. Consistency check by mass and/or energy conservation of appropriate quantities can be made, as well as consistency related to effect of boundary conditions on the numerical solutions [11].

For the calculation verification purpose, the most accurate and traditional way to quantitatively measure the error in a computational solution is by comparing the computational solution to a highly accurate solution. However, highly accurate solutions are known only for a relatively small number of simplified problems. It is also noted that when computational solutions are compared with highly accurate solutions, the comparisons should be examined along boundaries of interest or error norms should be computed over the entire solution domain. The accuracy of each of the dependent variables or functions of interest should be determined. The required fidelity of the numerical solution varies greatly with the type of solution variable that is computed.

3.1.4 Validation Process

The validation process is a process that develops confidence in the predictive capability of computational codes by comparing model results with real world phenomena in ranges of interest, while the verification process compares the computational model and with known mathematical solutions in the range of interest. Shown in Figure 3-4, the validation process includes comparing computation models and solutions with increasingly complex experimental or field data. Each physical parameter can be validated starting at the element or unit level (i.e., the scale at which an element has uniform properties and is exposed to uniform loads). This step can be used to improve the numerical capability of the tools or constitutive models. The next step after element-level validations is to validate tools and model performance at a benchmark tier (or tiers) using a slightly more complex experiment. Finally, system-level experimental tests can be performed and numerically replicated to validate the numerical code's predictive capability. A very well instrumented and well-documented field data can also be used for this purpose [12].

It is important to identify the physical processes and the associated phenomena elements and physics-based parameters for the validation. As stated in Section 2.2, PIRT method can be used to identify the physical phenomenon, describe how important the response of the phenomena is to the system, and detail the level of confidence in the current models.

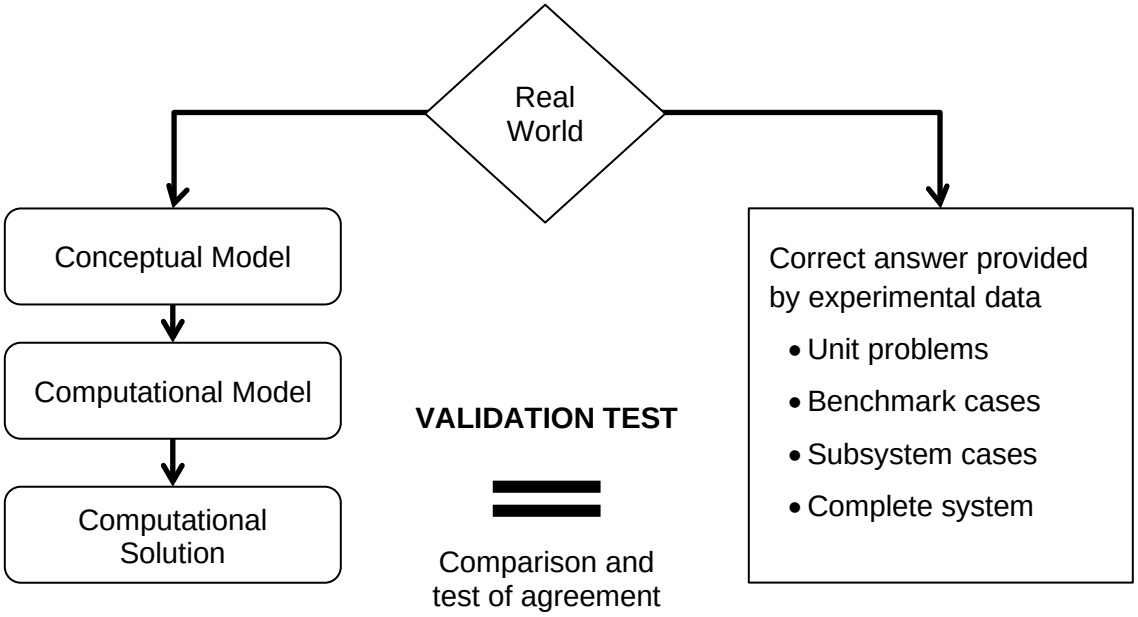

Figure 3-4: Schematic diagram of validation process [6]

The experimental data to provide correct answer for the validation can be categorized into four areas: unit problems, benchmark cases, subsystem cases and completes system cases [13]. It is noted that

information about uncertainties decreases as experiment becomes more complex, which is due to diversity of uncertainty source [14].

- For the unit problems, physics and geometry in experiments are simple and not coupled with other physics. Experiment will provide simple physical response data. Measured data need to include the entire model inputs, but uncertainties are given for all quantities.
- Benchmark cases uses special but non-functional facility with simplified geometry and properties. Boundary and initial conditions are used but they are very simple. Some coupling of physics is also possible. Uncertainties in the experiment are mostly given.
- Subsystem cases need functional and prototypical experimental facility. The experiment could be coupled with other subsystem level, but boundary and initial conditions are simple. Experimental facilities are rare and information about uncertainties is limited.
- For the system-level experiment, the facility is designed by using the actual system geometry and material properties. Physics are coupled completely, and boundary and initial conditions are also realistic. Very few facilities exist and little or no information about uncertainties is available.

Above four tiers can be also categorized into phenomenological, separate and integral effect tests. Phenomenological test problems are either simple or with closed-form analytic solutions to address a single code model or capability. Separate effects cases use tests that often model a particular component or geometry that is encountered in full-scale facilities, which are relatively simple experiments that address one or a few specific phenomena. Integral effects cases address experiments in facilities that are typically scaled models of a reactor plant. This can address many phenomena in the code, though within the limitations of the measurement data.

The fundamental concept of “Comparison and test of agreement” in validation cannot be easily defined; therefore, the basic idea can be captured from what methods and procedures of validation. It is very common to observe that the individual model is validated but the code is not. The main reason is that the validation need target experimental data to be compared, but not all phenomenological data are available especially for high-end technologies. In this case, code-to-code benchmark between other high fidelity codes, which is already validated, is a viable option.

In order to execute correct validation activity, following questions need to be implied [7].

- What is the method of measuring quantities in validation comparison?
- What is the method of determining the need of validation experiments?
- What are the characteristics of a validation experiments?

In the first question, the developer needs to address uncertainties in validation, both aleatory and epistemic.

3.1.4.1 Validation of CFD and Uncertainty

The uncertainties in scientific computing (aleatory and epistemic) are mainly due to the mathematical form of the model, and uncertainty analysis provides a procedure for including estimates of numerical error in the predictive uncertainty, which also occurred during validation activities. The developer needs to understand sources and characteristics of uncertainties and estimate uncertainties to provide higher-level of CFD validation.

Aleatory (random) uncertainties in model inputs are treated as random variables, while epistemic (lack of knowledge) uncertainties are treated as intervals with no assumed probability distributions [13]. Both uncertainties are unavoidable but epistemic uncertainties are reducible by improvement of mathematical models or can be predicted from stacking further knowledge (i.e., experiments), while aleatory uncertainty is irreducible. In CFD validation, epistemic uncertainty is generally considered.

The main sources of epistemic (or systematic) uncertainty are model inputs including system and boundary, numerical approximations and assumptions in correlations. Most common way to quantify uncertainties is to apply probability density function (PDF) or cumulative distribution function (CDF) to the model of interest, thus, propagation of the uncertainty. A CDF is the integral of the PDF from minus infinity up to the value of interest. The uncertainty quantification process is:

- Identification of uncertainty sources
- Characterization of uncertainties
- Estimation of uncertainties from numerical approximations
- Propagation of input uncertainties through the model

Uncertainty quantification methodology is well described in the LWRS RISA Pathway BEPU activity [15].

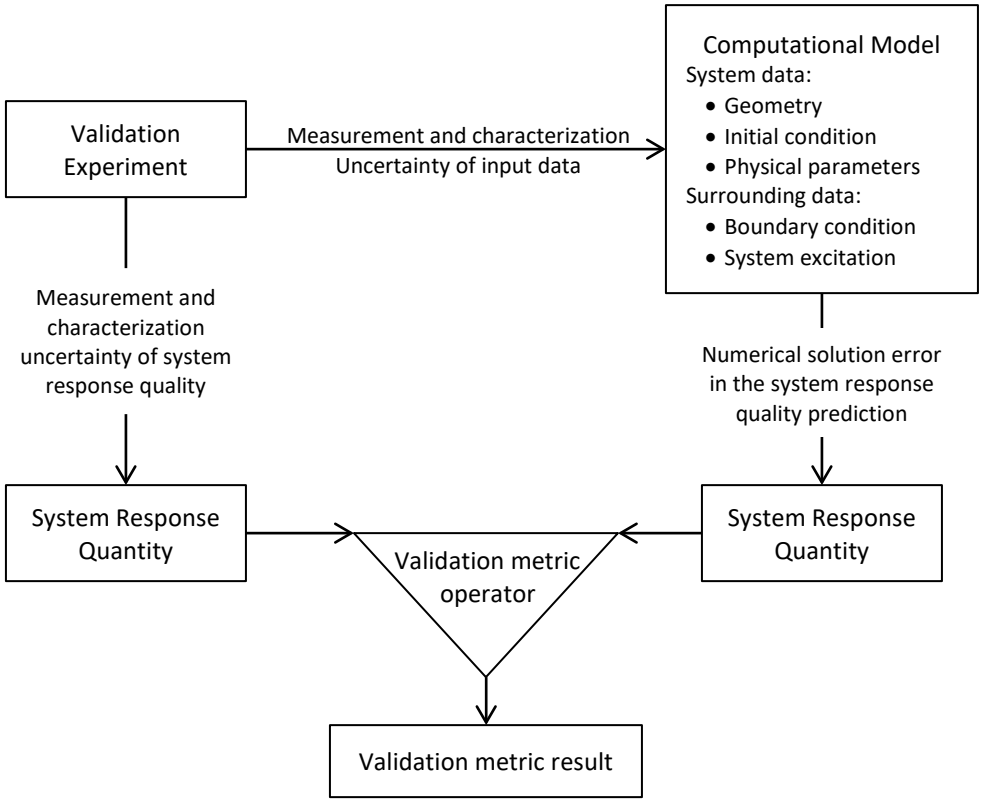

Figure 3-5: Validation metric flowchart considering uncertainties

A validation metric is a mathematical operator that requires two inputs: the experimental measurements of the system response quantification (SRQ) of interest and the prediction of the SRQ at the conditions used in the experimental measurements. A flowchart for computing a validation metric is given in Figure 3-5. In the ideal case, the validation metric is computed using specific validation experiments; however, more commonly one must use existing experimental data from the literature or from an industrial or laboratory database. A key part of computing the SRQ of interest is that the uncertainty in all model input parameters should be carefully measured during the experiment. Once the input uncertainties have been characterized, they are used as input to the model and are propagated to obtain the SRQ of interest (e.g., Figure of Merit: FOM). Depending on the nature of the model input uncertainties (purely aleatory, purely epistemic, or mixed), the SRQ is a precise CDF, an interval, or a p-

box, respectively. The key point that should be stressed with regard to the input uncertainties is that when they are propagated through the model, the model is expected to predict the experimentally measured variability in the SRQs that is due to the experimental variability of the inputs. Any disagreement between the experimentally measured and simulated SRQs (e.g., CDFs or intervals) is attributed to model form uncertainty, the source of which can be either physics modeling assumptions and/or imprecise knowledge of the input uncertainties.

It is noted that the concept of Figure 3-5 has a background that the experimental processes of nature are expected to reproduce by some degree by the mathematical model. However, the existence of experimental uncertainty in the measurement of the SRQ can potentially interfere with this seemingly reasonable expectation of the model. Even if bias errors in the measurements have been reduced to a negligible level, there is always random measurement uncertainty in the SRQ. However, uncertainties in measurement are not expected to be modeled, and there will be increased variance in the experimental results due to measurement uncertainty. Since this will not be exhibited in the model predictions, the model will be (incorrectly) assessed to be in error due to this effect.

3.2 Examples of V&V Approaches

3.2.1 Industry Standards (IEEE Std. 1012-2016)

IEEE Std. 1012-2016 [16] provides a V&V process standard that applies to all life cycle processes of software (*i.e.*, acquisition, supply, development, operation, maintenance). Particularly, to assure the independence of software V&V process this document proposes to establish *technical*, *managerial*, and *financial independence* as follows:

1. For *technical independence*, the independent V&V (IV&V) effort should be made by personnel who are not involved in the software development. In principle, any problems should be understood, formulated, and solved separately through V&V effort, which will help detect subtle errors that can be overlooked by those too close to solutions such as code developers. *Technical independence* also requires that V&V effort be made with its own set of tools for test/analysis separate from the developer's tools. Sharing tools is only limitedly allowed for computer support environments (e.g., compilers, assemblers, utilities) or for system simulations where an independent version would be too costly.
2. For *managerial independence*, the V&V responsibility should belong to an organization separate from the code development and program management organizations. *Managerial independence* also requires that V&V effort independently select the segments of the software and system to test/analyze, V&V techniques, schedule of V&V activities, and the specific technical issues/problems to work on. The results of IV&V effort should be reported in a timely fashion to both the development and program management organizations without any restrictions or adverse pressures.
3. For *financial independence*, the V&V budget should be vested in an organization separate from the software development organization. This is to prevent any situation where the V&V organization cannot complete its mission because of diverted funds or adverse financial pressures/influences.

Shown in Table 3-1, IEEE Std. 1012-2016 proposes five forms of independence (*i.e.*, classical, modified, integrated, internal, and embedded) are defined for the three independence parameters mentioned, which is to determine the degree of independence achieved through V&V process.

It is noted that the V&V process standard described in IEEE Std. 1012-2004 [16] is industry consensus and is largely endorsed by the documents for the software V&V released by NRC and DOE.

Table 3-1: Forms of IV&V described in IEEE Std. 1012-2004 [16]

IV&V Form	Technical	Management	Financial
Classical	I	I	I
Modified	I	i	I
Integrated	i	I	I
Internal	i	i	i
Embedded	e	e	e
Note: I=Rigorous independence; i=Conditional independence; e=Minimal independence			

3.2.2 NRC Regulatory Guide 1.168

The NRC's regulatory guide 1.168 provides guidance to the staff of U.S. NRC with respect to verification, validation, reviews, and audits for computer software used in safety systems of NPPs [17]. Since the RISA Toolkit aims to be used for safety-related NPP system simulations, the toolkit developers as well as independent V&V performer and reviewer should consider this regulation to support future license acquirement. Originated from the IEEE Standard and in compliance with 10 CFR Part 50, the latest version of RG 1.168 is revision 2, published in July 2013.

Based on the 10 CFR Part 50 Appendix A and B, software V&V, review and audit process should require development of appropriate quality assurance program. These requirements include:

- Criterion I, “Organization,” specifies that applicants must (1) assure that an appropriate quality assurance program is established and effectively executed, and (2) verify (for example, by checking, auditing, and inspection) that activities affecting safety-related functions have been correctly performed.
- Criterion II, “Quality Assurance Program,” requires, in part, that activities affecting quality be accomplished under suitably controlled conditions. Controlled conditions include the use of appropriate equipment, suitable environmental conditions for accomplishing the activity, and assurance that all prerequisites for the given activity have been satisfied. The criterion also requires that the program consider the need for verification of quality by inspection and test.
- Criterion III, “Design Control,” requires, in part, that design control measures provide for verifying or checking the adequacy of design.
- Criterion XI, “Test Control,” requires, in part, that a test program be established to assure that all testing required to demonstrate that structures, systems, and components will perform satisfactorily in service is identified and performed in accordance with written test procedures, which incorporate the requirements, and acceptance limits contained in applicable design documents.
- Criterion XVII, “Quality Assurance Records,” requires, in part, retention of records to furnish evidence of activities affecting quality. The records shall include identification and descriptions of actions taken in connection with any changes or design deficiencies noted.
- Criterion XVIII, “Audits,” requires, in part, that a comprehensive system of planned and periodic audits be carried out to verify compliance with all aspects of the quality assurance program and to determine the effectiveness of the program.

These six criteria should be taken into account for development, upgrade, V&V activities, software maintenance and quality assurance to ensure software integrity and reliability, which will improve product credibility and facilitate industry deployment.

3.2.3 Federal Standards and DOE Guide

The 10 CFR 830 Subpart A describes the requirements of quality assurance program (QAP) for the DOE nuclear facilities and activities, which is supplemented by DOE O 414.1C [18]. DOE G 414.1-4 [19] is also a software guide for use with both 10 CFR 830 Subpart A and DOE O 414.C, providing instructional guidance to be applied with the requirements specified in DOE O 414.C.

As for the independence of software V&V or QAP, 10 CFR 830 Subpart A and DOE G 414.1-4 include the following descriptions:

- 10 CFR 830 Subpart A, Criterion 10

  "Establish sufficient authority, and freedom from line management, for the group performing independent assessments."

- DOE G 414.1-4

  "SQA (software quality assurance) Evaluator - an independent reviewer of the computer software, who is not affiliated with the software developing organization."

  "Independence between the evaluator and the sponsor is critical for completion of a formal SQA evaluation and should be maintained throughout the Central Registry submittal process."

ASME NQA-1 [20], a guidance for implementing federal regulations associated with QA described in 10 CFR 50 Appendix B, also share the fundamental principles for the role and responsibility of QA organization as follows:

"First and foremost, the QA team must be able to function independently from the organizations it is responsible for overseeing. This includes having the authority to stop work or bring an issue independently "up the chain" to the site manager or other top executive as defined by the facility's governance model."

3.3 Software Quality Assurance Program in INL

The Idaho National Laboratory has a laboratory-wide procedure designated to SQA and designed to be used by all employees who propose, develop, modify, acquire, administer, maintain, or retire computer software, unless specifically excluded. This program includes SQA and V&V procedures covering all phases of software development activities. It is noted that documents listed in this section are not publicly available. However, INL's SQA is compliant with the ASME NQA-1 [20].

The INL QA program requires all software applications to have "Safety Software Determination (SSD)" and "Quality Level Determination (QLD)" in order to determine whether it is safety or non-safety and quality level (QL-1, QL-2 and QL-3). Once that is determined, the applicant can prepare related documents for quality level application as well as planning and performing V&V.

There is no consistent definition for the term "quality level". The QL only serves as the designator to identify the unmitigated risk or potential consequence level associated with the failure of an item or activity and facilitates communication for a common understanding of the rigor to be applied through the appropriate implementation procedures. The INL SQA program proposes the following guidance for QLs:

- Quality level 1 (QL-1) equates to high-unmitigated risk or high potential consequence level of failure.
- Quality level 2 (QL-2) equates to medium-unmitigated risk or medium potential consequence level of failure.

- Quality level 3 (QL-3) equates to low unmitigated risk.
- Quality level 4 (QL-4) equates to a no risk item or service.

The lists of applicable INL documents, templates, tools and forms is shown below. It is noted that the following documents are not publicly available.

Program Description
- PDD-13610, "Software Quality Assurance Program"

Program Implementation Procedures
- LWP-13620, "Managing Information Technology Assets"
- LWP-20000-01, "Conduct of Research Plan"

Templates
- TEM-135, "IT System Requirements Specification"
- TEM-140, "IT Software Design Description"
- TEM-141, "Software Test Plan"
- TEM-142, "Software Quality Assurance Plan"
- TEM-143, "Configuration Management Plan"
- TEM-162, "IT Project Management Plan"
- TEM-214, "IT System RTM"
- TEM-215, "IT Asset Maintenance Plan"
- TEM-216, "Software Verification and Validation Plan"

Tools
- Enterprise Architecture Repository / Capabilities & Technology Management (CTM)
- QLD and SSD applications
- Safety Software Listing in CTM

Forms
- Form 414.A89, "Quality Level Determination Risk Analysis Method"
- Form 414.A91, "Quality Level Determination Safety Software Analysis Method"
- Form 562.29, "Software Product Review Report and Checklist"
- Form 562.33, "INL SQA Assessment Checklist"
- Form 562.37, "Safety Software Determination"
- Form 562.38, "Safety Software Training Documentation Form"
- Form 562.40, "IT Asset Management Responsibilities Checklist"

Application of above document is shown in Figure 3-6 (Appendix E of LWP- 13620, "Managing Information Technology Assets"). The RISA Toolkit developer should first review PDD-13610 and LWP-13620 to understand the procedure of improving quality and set QLD and software V&V plan for future QA application.

Providing documents listed above will improve credibility of the RISA Toolkit and eventually facilitate industry deployment of risk-informed tools and methods.

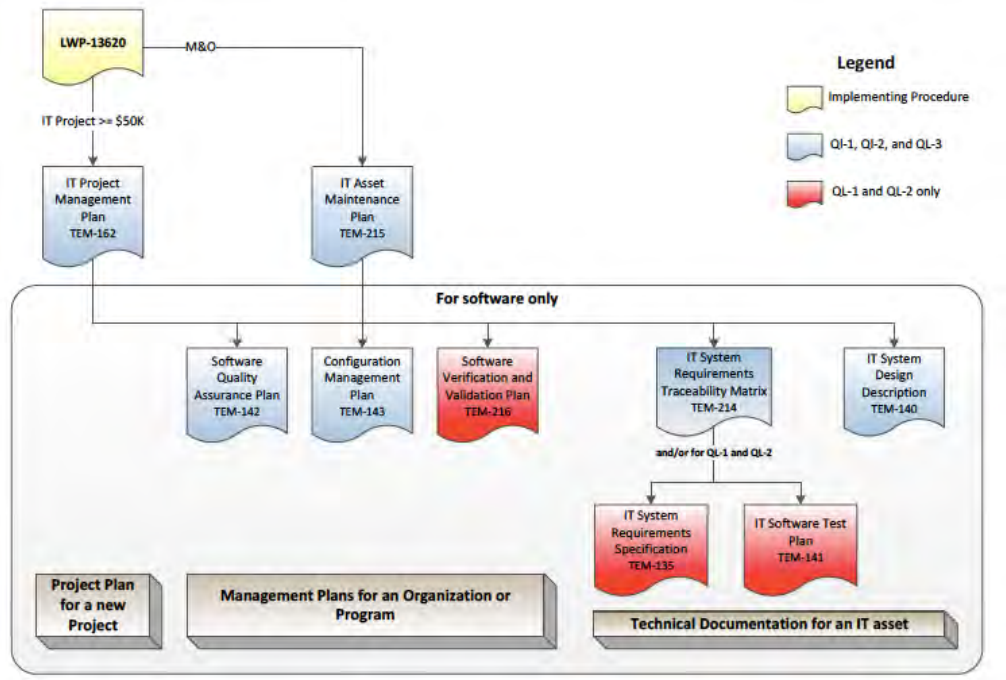

Figure 3-6: IT Asset document flow diagram

4. TECHNOLOGY MATURITY ASSESSMENT OF MOOSE-BASED TOOLS

4.1 MASTODON

4.1.1 Overview

Multi-hazard Analysis for STOchastic time-DOmaiN phenomena (MASTODON) is a open-source finite element application that aims at analyzing the response of 3-D soil-structure systems to natural and man-made hazards. MASTODON currently focuses on the simulation of seismic events and has the capability to perform extensive "source-to-site" simulations including earthquake fault rupture, nonlinear wave propagation and nonlinear soil-structure interaction (NLSSI) analysis. MASTODON can also perform probabilistic risk assessment that includes stochastic analyses as well as post processing the results to calculate fragilities and performing fault tree and event tree analysis for risk assessment [21].

MASTODON was originally developed under the guidance of RISMC Pathway in 2016 for external hazards analysis [22] including experimental plan to support V&V of the code [23] and multi-hazard tool V&V plan [24]. In 2017, Beta 1.0 version was released including seismically-induced fire PRA methodology [25], and its performance was demonstrated through NLSSI analysis project along with DOE/NNSA [26].

4.1.2 Features

MASTODON is a MOOSE-based application which performs finite element analysis of dynamics of solids, mechanics of interfaces, and porous media flow. It is equipped with effective stress space nonlinear hysteretic soil constitutive model (I-soil) as well as structural materials such as reinforced concrete. It includes interface models that simulate gapping, sliding and uplift at the interfaces of solid media such as the foundation-soil interface of structures. Absorbing boundary models for the simulation of infinite or semi-infinite domains, fault rupture model for seismic source simulation, and the domain reduction method for the input of complex, three-dimensional wave fields are incorporated. Since MASTODON is a MOOSE-based application, it can efficiently solve problems using standard workstations or very large high performance computers.

The major capabilities of MASTODON are:

- Source-to-site simulation
- Nonlinear site-response and soil-structure interaction
- Seismic probabilistic risk assessment (SPRA)
- Implicit and explicit integration
- Easy coupling with other physics tools
- Highly parallelizable (MOOSE tools routinely used on thousands of computer processors)
- Automated testing and documentation maintenance

MOOSE contains several physics modules (so-called kernel), each of which provides materials, boundary conditions, constraints, and other tools for a certain class of physics, such as mechanics, heat conduction, fluid flow, etc. Of these modules, MASTODON includes tensor mechanics, contact, and stochastic tools modules, which provide capabilities of solid mechanics and dynamics, contact interface simulation, and uncertainty quantification, respectively. MASTODON also includes another MOOSE application called Black Bear, which simulates the degradation of structural materials like concrete and steel from aging-related phenomena (e.g., alkali-silica reaction). In addition to these modules and applications, MASTODON includes other tools that are specific to seismic analyses, such as nonlinear soil materials, seismic protective systems, fault rupture, absorbing boundaries, etc.

MASTODON has versatility of pre- and post-processing of the simulation. One of eminent features is mesh generation from an image file. For example, in simulations where soil layers are nonhorizontal and nonplanar, MASTODON can process image files (i.e., JPEG, PNG, etc.) of the soil profile and create a soil layer mesh, where material blocks are based on colors in the image. For creating 3-D soil domains, multiple 2-D images with soil profiles at different 2-D cross sections of the soil domain can be provided as an input. MASTODON can also automatically ensure that the soil mesh density is adequate for wave propagation of a certain frequency. Figure 4-1 is an example of auto-generated mesh from an image.

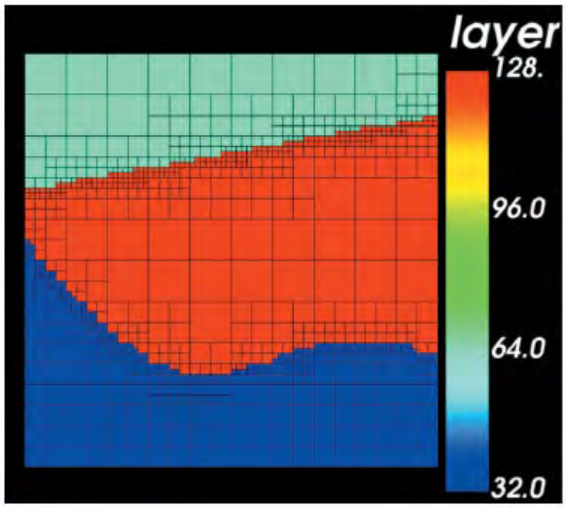

Figure 4-1: Example of auto-generated mesh from image

4.1.3 Verification and Validation Status

MASTODON uses INL's Software Quality Assurance Program (SQAP) of MOOSE and MOOSE-based application document to set V&V plan, which is in compliance with ASME NQA-1. However, this document is not publicly available.

From the stage of code planning and development, MASTODON team proposed a V&V plan including various seismic and flooding experiments [23 and 24]. However, not all scheduled V&V activities were accomplished. Table 4-1 shows available completed verification problems of MASTODON.

Table 4-1: List of available completed verification problems of MASTODON

Problem Title	Category
Newmark time integration model	Function verification
Hilber-Hughes-Taylor time integration model	Function verification
Analytical verification of the I-Soil constitutive model • Data file • Darendeli material model • GQ/H (General Quadratic/Hyperbolic) material model • Pressure dependency model (strength and/or stiffness)	Function verification Model verification Model verification Model verification
Tensor mechanics model	

Problem Title	Category
• Small strain Timoshenko beam bending model	Model verification
• Small strain Euler beam bending model	Model verification
• Small strain Euler beam axial loading model	Model verification
• Large strain/large rotation of cantilever beam model	Model verification
• Torsion model	Model verification
• Small strain Euler beam vibration model	Model verification
• Small strain Timoshenko beam vibration model	Model verification
• Small strain massless beam vibration with a lumped mass model	Model verification
• Small strain massless beam damped vibration with a lumped mass having rotational moment of inertia model	Model verification

Very few MASTODON validation activities are reported due to availability of experimental facility and data. Code-to-code benchmark was performed for nonlinear soil-structure interaction problems, seismic isolation, vibration response of structural elements such as beams and shells, etc. Codes including LS-DYNA and ABAQUS, which are general-purpose finite element simulation tools, and OpenSees, which is an open-source seismic analysis code, were used to compare the results. As of September 2021, one experimental research is currently progressing, and the data will be used for MASTODON validation: experiments on small- and medium-scale models of concrete foundations placed on soil and subjected to static and dynamic loadings. Other validated capabilities include (1) wave propagation in soil, which has been validated against results from a large-scale geotechnical laminar box experiment, (2) seismic isolator materials, which were benchmarked against validated codes, and (3) seismic-induced fluid-structure interaction in tanks, which involved comparison with experimental data [36].

4.1.4 Technical Maturity Assessment

This section describes MASTODON technical maturity assessment results based on the requirements as the RISA Toolkit.

Development Level

MASTODON code was first developed under the LWRS program to simulate beyond design basis accident level natural hazard such as earthquake and tsunami as well as fire and structural integrity analysis, issues that occurred in the Fukushima 2011 event. Since MASTODON is a MOOSE-based code, it has all the benefits from the MOOSE framework. Using MOOSE kernels, MASTODON solves partial differential equations of dynamic motion equation or mechanical wave equations. The main boundary conditions are displacement and traction of target systems. MASTODON can simulate earthquake fault rupture, source-to-site wave propagation during an earthquake, nonlinear site-response and soil-structure interaction stochastic analysis and perform seismic PRA. Most of key physical models are already developed in verified (see Table 4-1). However, validation of the model and code is still incomplete.

MASTODON can be used with MOOSE GUI Peacock (https://mooseframework.inl.gov/application_usage/peacock.html), Gmsh (https://gmsh.info/), CUBIT (https://cubit.sandia.gov/), Telis for mesh generation and pre-processing. ParaView (https://www.paraview.org/) and Peacock can be used for post-processing and graphic generation.

The code has open-source policy and development is managed in GitHub repository hosting service. Manuals, examples and other documents are available at MASTODON webpage [21]. MASTODON is available for Linux, MacOS and limited support on Windows.

Use of Proven Technology

MASTODON uses PETSc nonlinear solver package and libmesh to provide the finite element discretization. These libraries allow use preconditioned Jacobian-Free-Newton-Krylov (PJFNK) method to maximize computational effectiveness for solving nonlinear analysis. For linear analyses, Newton method is used to ease convergence. For seismic analysis, MASTODON can use implicit integration schemes such as Newmark-β and Hilbert-Hughes-Taylor (HHT) algorithms, as well as the explicit central difference algorithm, which are commonly used integration algorithms for wave propagation applications. These technologies are mature and fully verified and validated.

Supporting software for post- and pre-processing are also showing high level of maturity. MASTODON is most compatible with Exodus-II and the Gmsh ASCII format for input data and mesh. To generate mesh, CUBIT (free for U.S. government users) and Trelis51 (commercial version of CUBIT) are widely used. ParaView, which is widely used software with an active use base can be used for post-processing. However, performance of the Peacock has not been evaluated under the RISA Pathway project.

PRA Capability/Applicability

MASTODON has a built-in seismic PRA capability through either the intensity-based assessment or time-based assessment. Intensity-based assessment involves calculating the seismic risk for the specific hazard intensity (expressed as peak ground acceleration or a spectral acceleration at a certain period). Time-based assessment involves repeating the intensity-based risk calculation across a wide range of hazard intensities in the seismic hazard curve and calculating the sum of these risks.

Figure 4-2 is a schematic diagram of seismic PRA in MASTODON. The process can be automated by taking fault trees, event trees, seismic hazard curve, hazard consistent ground motions, and the finite element model as inputs. Monte Carlo or Latin Hypercube sampling method can be used for probability calculation.

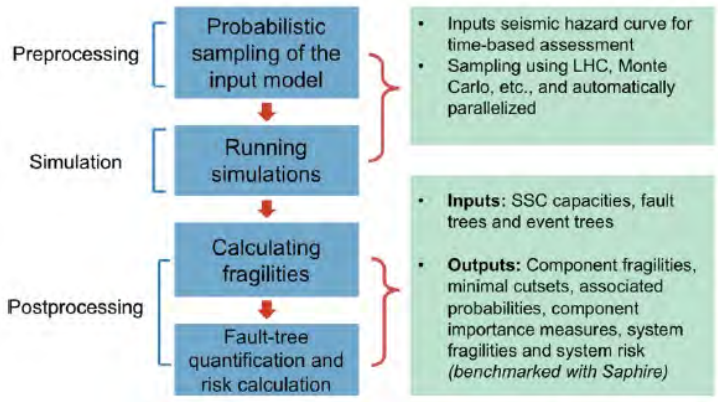

Figure 4-2: Schematic diagram of seismic PRA using MASTODON

MASTODON also includes event tree and fault tree analysis, which have been implemented both as a standalone Python library as well as in C++.

Documentation

MASTODON uses open-source binary Doxygen to generate on-line document and LaTeX script. Following documents are available at official webpage [21].

Following documents are available:

- User manual
- Theory manual
- Developer manual
- Input syntax description
- Run examples
- Tutorial
- Installation and running guide
- SQA documents
- V&V suites

System Requirements

MASTODON is compatible with Linux, MacOS and limited support of Mirosoft Windows.

Most preferable operating system is Linux and MacOS environment. Minimum system requirements for MOOSE-based tools are:

- GCC/Clang C++14 compliant compiler (GCC @ 5.1.0, Clang @ 3.5.1 or greater)
- Memory: 16 GBs (debug builds)
- Processor: 64-bit x86
- Disk: 30GB

It is noted that Intel compilers are not supported.

Various Linux OS are compatible: CentOS, Fedora, OpenSUSE and Debian (Ubuntu, Mint).

MOOSE-based codes cannot be installed directly in Windows systems, hence, a user would need to install Linux environment under Windows such as Ubuntu 20.04. VcXrv is Windows X server which is necessary to use Peacock GUI.

MASTODON also allows installation and use under the INL HPC system.

Official computer OS comparison report is available upon request.

Easy Installation

Installation, update and uninstallation guideline for different operating systems is described on the MASTODON webpage. Installation of MOOSE libraries is necessary prior to installation of MASTODON. The software can be pulled from the GitHub repository. No specific issues were found in installation.

Running syntax and tutorial with sample files are included in the code.

Graphic User Interface

MASTODON can be executed under the MOOSE-based GUI Peacock for both pre- and post-processing. Peacock can create the input file and visualize mesh, mesh blocks, boundaries, nodes, and then can execute MASTODON to generate output data and graphical images. Peacock also has capability to verify input, mesh and output to review code response histories, stress contours and response spectrum. Figure 4-3 shows various pre- and post-processing set up for MASTODON. For mesh generation, CUBIT and Trelis are mostly used and Peacock and MeshGenerator of MOOSE are the also available options

including automatic mesh generation from an image. Graphical results can be visualized with Peacock, ParaView, and VectorPostprocessor system of MOOSE.

Figure 4-3: Pre- and post-processing setup for MASTODON

Version Control

MASTODON uses GitHub repository to maintain automatic version control for each change and version control information such as change history and contributions can be tracked on GitHub.

Official version comparison report is available upon request.

V&V Activity/History

Verification activities are summarized on the webpage. The list of completed verification tasks is shown in Table 4-1. The failure analysis report shows the list of issues and resolutions.

As of September 2021, no official validation report is available. Validation of specific capabilities are published in various documents but are not compiled into a single report.

QA Program

Quality of MASTODON is controlled under the INL's SQA program, which is in compliance with ASME NQA-1. The entire MOOSE-based toolset follows the same quality standard document "Software Quality Assurance Program of MOOSE and MOOSE-based Tools" (internal document). This activity includes documentation platform MooseDocs and verification testing tool CIVET.

MOOSE and MOOSE-based applications are designated as QL-2 non-safety software but are managed to implement QL-1 safety software requirements. Generally, seismic analysis and risk assessment software are required to meet the NQA-1 standard in order to be used for the design and risk assessment of nuclear safety-critical facilities.

Following SQA documents are available upon request:

- Software Quality Assurance Plan for MOOSE and MOOSE-based applications, including
  - Configuration Management Plan (CMP)
  - Verification and Validation Plan (VVP)
  - Software Quality Plan (SQP)
- Enterprise Architecture (EA) Record
- Safety Software Determination (SSD)
- Quality Level Determination (QLD)
- Software Design Description (SDD)

- System Requirement Specification (SRS) and RTM

Software design descriptions, requirement specifications on code functions and corresponding RTM is recorded in the GitHub repository.

As of September 2021, no independent Asset Management Plan (AMP) is available.

Webpage

Official webpage is https://mooseframework.inl.gov/mastodon. This site includes basic information of the code, manuals, installation guides, examples and contact information for permission to use. All INL controlled MOOSE-based tools are in same host directory. It is noted that the webpage is hosted by INL and accessibility could be limited due to host stability.

GitHub repository for MASTODON is https://github.com/idaholab/mastodon.

User Support

Frequently Asked Questions are available at MASTODON webpage.

MASTODON user group https://groups.google.com/forum/#!forum/mastodon-users is recommended for contact development team.

User support and communication is established in GitHub repository https://github.com/idaholab/mastodon/issues.

Failure analysis report in MASTODON webpage (https://mooseframework.inl.gov/mastodon/sqa/mastodon_far.html) is the list of issues reported by user and their resolutions suggested by development team through GitHub repository.

Training Program

No regular training program is available. Training is available upon request.

Examples of MASTODON runs are available at webpage with input files in GitHib repository. Each example has problem, model, input, output and result description. Available example runs are listed below.

- Single element direct simple shear test with auto-generated backbone curve using Darendeli modulus reduction curves (implicit and explicit time integration cases).
- Single element direct simple shear test with auto-generated backbone curve using the GQ/H formulation of Groholski (implicit and explicit time integration cases).
- Nonlinear site-response analysis (implicit and explicit time integration cases).
- Frictional contact using a user-defined backbone curve in I-Soil.
- Dynamic analysis of a simplified model of a nuclear power plant structure.
- Dynamic response of a simplified 1D model of nuclear power plant with a mesh generated using BeamMeshGenerator.
- Displacement-controlled frictional contact problem with elastic soil.
- Displacement-controlled frictional contact problem with I-soil and using the restart option for static initialization.
- Force-controlled contact problem.
- Demonstration of domain reduction method.
- Seismic risk assessment of a generic nuclear facility using the MASTODON FTA Python module.
- Soil-structure interaction analysis of an undamped rigid structure on undamped linear soil.

- Cantilever example demonstrating central difference time integration.
- Cantilever example demonstrating central difference time integration using HEX20 elements.
- Cantilever example demonstrating mesh requirements for accuracy in bending of static and dynamic simulations.
- Mesh refinement using Marker and Indicator, and demonstration of SidesetMoment and AverageValue postprocessors.
- Fluid-structure interaction analysis of a tank subjected to uni-directional harmonic acceleration.
- Fluid-structure interaction analysis of a tank subjected to uni-directional earthquake acceleration.
- Experimental validation of fluid-structure interaction simulations.
- Soil-Structure Interaction and Fluid-Structure Interaction in nuclear power plant structure.
- Seismic analysis of a base-isolated nuclear power plant building.

License

Copyright of MASTODON belongs to Battelle Energy Alliance, LLC. However, MASTODON is an open-source software managed under the terms of the GNU Lesser General Public License as published by the Free Software Foundation.

User will receive a copy of the GNU Lesser General Public License along the software source code. Or can request a copy at the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

4.1.5 Conclusions and Remarks

MASTODON was initiated for the natural hazard analyses under the LWRS Program. Since it uses MOOSE framework, development procedure, documentation, and QA processes are standardized and are compliant to NQA-1 standards. The project aims to bring these state-of-the art methods and tools for design, analysis, and risk assessment under one software framework and to make them more accessible to engineers and critical infrastructure stakeholders. MASTODON is under active development, and it is currently capable of structural mechanics and dynamics, contact mechanics, fluid-structure interaction stochastic simulations, fragility analyses, and fault tree and event tree analyses along with various GUI applications. Technical maturity assessment results are shown in Table 4-2.

As a RISA Toolkit, MASTODON shows good benefits since it has built-in seismic PRA capability. Since MASTODON can also handle structure integrity during severe natural hazard events (e.g., external flood), it can be used in the RISA Pathway projects including issues arisen from global climate change. In terms of RISA Pathway activity, following remarks are addressed.

- Official validation activity tests and corresponding documentation is necessary
- Additional demonstration study on coupling with other physics-based codes can improve code credibility
- Generally, code has enough technical maturity to be used in the RISA Pathway.

Table 4-2 MASTODON technology maturity assessment result

Requirements	Importance	Description	TRL
Development level	High	Code development is mostly completed. Improvement is planned for computational enhancement. GUI and pre- and post-processing methodologies are also set. Manuals, SQA and related documents are ready. More demonstration including coupling with other physics tools is necessary. Need official validation activity.	6
Use of proven technology	High	Mature technologies used in MASTODON. No specific issue was found.	9
PRA capability/applicability	High	MASTODON has built-in seismic PRA capability. However, demonstration and validation are needed.	6
Documentation	High	Manual or other supporting documents are available in official webpage. However, some part of webpage remains incomplete.	5
System requirements	Low	Windows version is not fully compatible, but no issue for running the code. Suggest performance test study in different OS.	7
Easy installation	Medium	Installation used GitHub pulling. MOOSE libraries need to be pre-installed. No specific issue was found.	9
Graphic user interface	High	Several of GUI software is available and tested. Peacock need performance test. No specific issue was found.	8
Version control	Low	GitHub maintains version. No specific version control report is available.	5
V&V activity/history	High	Verification report is available. No official validation report is available.	3
QA program	Medium	SQA documents are available. Some of QL-1 documents are also available.	8
Webpage	High	Official webpage at MOOSE framework. However, some part of webpage remains incomplete.	5
User support	High	Developer supports directly to the issue through GitHub.	9
Training program	Low	Development team organizes training program as needed. Training tutorials and examples are available with documentation.	9
License	Low	MASTODON is open-source.	9

4.2 BISON

4.2.1 Overview

BISON is a finite element-based nuclear fuel performance analysis code applicable to a variety of fuel forms including light water reactor fuel rods, tristructural isotropic (TRISO) particle fuel, and metallic rod and plate fuel. It solves the fully-coupled equations of thermomechanics and species diffusion, for either 1D spherical, 1D axisymmetric, 2D axisymmetric, 2D Cartesian or 3D geometries. Fuel models are included to describe temperature and burnup dependent thermal properties, fission product swelling, densification, thermal and irradiation creep, fracture, and fission gas production and release. Plasticity, irradiation growth, and thermal and irradiation creep models are implemented for clad materials. Models are also available to simulate gap heat transfer, mechanical contact, and the evolution of the gap/plenum pressure with plenum volume, gas temperature, and fission gas addition. BISON is based on the MOOSE framework and can therefore efficiently solve problems using standard workstations or very large high performance computers [37].

BISON is one of the first MOOSE-based tools developed in INL in conjunction with other US national laboratories, with the first efforts initiated in 2008 [38]. First demonstration studies were formed for LWR and TRISO coated fuels [39]. For the LWRS program, BISON was used in several projects under the Materials and RISA Pathways. In 2014, application of BISON to accident tolerant fuel (ATF) and coupling capability with risk-informed tools (e.g., DAKOTA and RAVEN) was first reviewed [40]. Application to the fast reactor fuel performance was also reviewed [41]. Most recently, RISA risk-informed enhanced resilient power plant project used BISON for fuel performance during beyond design basis accident analyses [42].

4.2.2 Features

BISON is a MOOSE-based application and it performs finite element analysis of the nuclear fuel performance by using a multi-physics solver provided by the MOOSE framework. This provides solutions for spatial and temporal variations in the variables that are being solved for. The code also has been demonstrated for TRISO fuel and plate metal fuel. Since the code operates under the MOOSE framework, it is easy to couple with other MOOSE-based tools and high parallelizable running is possible with multiple computer processors.

The major capabilities of BISON focused on traditional LWR fuel applications are as follows:

- Simulation of oxide fuel behavior considering
  - Material property based on temperature, burnup and porosity
  - Volumetric heat generation
  - Thermal and fission product swelling, and densification strains
  - Thermal and irradiation creep
  - Fuel fracture via relocation and smeared cracking
  - Fission gas release model in transient, grain growth and sweeping, and athermal
- Simulation of fuel gap and plenum considering
  - Gap heat transfer
  - Mechanical contact
  - Plenum pressure as a function of evolving mechanical gas volume, gas mixture, and gas temperature approximation
- Simulation of cladding behavior considering
  - Thermal and irradiation creep

  - Thermal expansion
  - Irradiation growth
  - Plasticity
  - Hydride damage
- Simulation of coolant channel considering closed channel thermal-hydraulics with heat transfer coefficients

Capabilities focusing on other type of fuel are not shown in this report. More information is available on the BISON webpage [37].

4.2.3 Verification and Validation Status

BISON uses INL's Software Quality Assurance Program (SQAP) for MOOSE and MOOSE-Based Application document to define its V&V plan, which is in compliance with ASME NQA-1 requirements. However, this document is not publicly available.

The BISON development team researched specific V&V methodologies for BISON for different fuel types including regression and defect tests [43]. An extensive verification activity using analytic and manufactured solutions was conducted recently [44]. A total of 22 verification problems are assessed to confirm conservation equations to solve heat conduction and mechanics of the BISON code including:

- Basic heat conduction problem
  - Plate with internal heating
  - Plate with temperature dependent thermal conductivity
  - Plate with temperature dependent thermal conductivity and internal heating
  - Rectangular adiabatic plate
  - Hollow cylinder with Dirichlet boundary conditions
  - Hollow cylinder with temperature dependent thermal conductivity
  - Hollow cylinder with temperature dependent thermal conductivity and internal heating
  - Hollow cylinder with internal heating and outside convection boundary
  - Hollow cylinder with internal heating and inside convection boundary
  - Short solid cylinder
  - Solid sphere with internal heating
  - Spherical shell with Dirichlet conditions
  - Spherical shell with temperature dependent thermal conductivity
  - Solid sphere with spatially dependent internal heating
  - Solid sphere with internal heating and convective boundary condition
  - Spherical shell with internal heating and inside heat flux condition
  - Method of manufactured solutions (MMS) for one-dimensional conduction
  - MMS for two-dimensional conduction
  - MMS for transient conduction
- Gap heat transfer
  - Cartesian gap heat transfer
  - Cylindrical gap heat transfer
  - Spherical gap heat transfer

BISON has been validated for LWR, TRISO, Metallic, Nitiride, Carbide, and Mixed Oxide (MOX) fuels. For the LWR fuels, the BISON development team developed a specific validation methodology and covered wide range of validation with experiments [45]. Following are the list of BISON validation cases dedicated to separate effects, integral fuel rod experiments and accident behavior of LWR fuels.

Calvert Cliffs 1 Test Rods
IFA 431 Rods 1 2 and 3
IFA 432 Rods 1 2 and 3
IFA 515 10 Rod A 1
IFA 519 Rod DH and Rod DK
IFA 534 Rods 18 and 19
IFA 535 5 6 Rods 809 812
IFA 562 Rods 15 17
IFA 597 3 Rod 7 and Rod 8
IFA 636 2 Rod 5
IFA 677 1 Rod 1
IFA 681 Rods 1 2 and 3
LOCA ANL Cladding Burst Tests
LOCA Hardy Tube Test
LOCA IFA 650 10
LOCA IFA 650 2
LOCA IFA 650 4
LOCA IFA 650 9
LOCA MT 4 and MT 6 A
LOCA ORNL Burst Tests
LOCA PUZRY
LOCA REBEKA
Riso AN 2
Riso AN 3
Riso AN 4
Riso AN 8
Riso 2 GE m STR 013
OSIRIS J 12 5
LOCA Studsvik Rods 191 and 196
RE Ginna Rodlet 2 and Rodlet 4
REGATE
RIA CABRI REP Na 2 3 5 10
RIA CABRI REP Na 4
RIA NSRR FK Tests
Riso 2 GE m STR 013
Riso AN 2
Riso AN 3
Riso AN 4
Riso AN 8
Riso GE 7 Fuel Pin ZX 115
Riso II 3
Riso II 5
Super Ramp
TRIBULATION BN 1 3 BN 1 4 and BN 3 15
US PWR 16 x 16 Rods TSQ 002 and TSQ 022
Zirconium Hydrides Experiments
LOCA Studsvik Rods 191 and 196
OSIRIS H 09
OSIRIS J 12 5
RE Ginna Rodlet 2 and Rodlet 4
REGATE
RIA CABRI REP Na 2 3 5 10
RIA CABRI REP Na 4
RIA NSRR FK Tests
OSIRIS H 09
High Burnup Effects Program (HBEP) A1/8-4 A3/6-4 and H8/36-4
High Burnup Effects Program (HBEP) BK 363, BK 365 and BK 370

Additional efforts were given to participate international benchmark programs such as:

- AREVA Idealized Case
- Gap Conductance
- IAEA FUMEX XII
- OECD/NEA/WGFS RIA Fuel Codes Benchmark Case 5
- Detail of validation activities is available at BISON webpage [37].

4.2.4 Technical Maturity Assessment

The following information describes BISON technical maturity assessment results based on the requirements as the RISA Toolkit.

Development level

BISON is one of most mature MOOSE-based tools in terms of development, V&V, demonstration, and recognition by the nuclear industry. Similar to other MOOSE-based tools, BISON has the benefit of the MOOSE framework such as coupling flexibility, GUI, parallel execution, Doxygen, etc. Especially, numerical models for LWR fuel performance and cladding behavior are widely demonstrated and validated.

The webpage for the MOOSE framework has a large amount of information about installation, theory, tutorial and other necessary documents. The SQA program is also well maintained.

The BISON input and mesh can be generated for simple problems using internal tools, using a Python script provided as part of the BISON distribution that uses Cubit to generate a mesh, or by using other external software. For the post-processing and visualization of result, BISON is compatible ParaView, Ensight, Peacock, Blot, Patran, and Visit software.

BISON source code is maintained on a GitLab repository maintained on an INL High Performance Computing (HPC) system. Users need a licensing agreement with INL to access BISON. This code can be installed on local computers or HPC clusters.

Use of proven technology

Similar to MASTODON, BISON uses MOOSE kernels to solve relevant physics. The PETSc nonlinear solver package and libMesh are used for the finite element discretization. These libraries allow using the PJFNK method to maximize computational effectiveness for solving nonlinear analysis. The Elsentatd-Walter algorithm is applied when required liner tolerance in Newton solve. For linear analysis Newton method is used to ease convergence. These technologies are mature and fully verified and validated.

Supporting software for post- and pre-processing are also showing high level of maturity. However, performance of the Peacock has not been evaluated under the RISA Pathway project.

PRA capability/applicability

BISON has built-in Monte Carlo and Latin Hyper-Cube (LHS) sampling capabilities. Other option is to use of the stochastic tool module in MOOSE framework. External PRA tools such as MOOSE-based RAVEN could be coupled along with CUBIT [49].

Documentation

BISON uses an automated system that uses a combination of information in the source code and Markdown files to generate documentation that is consistent with the current state of the code. The following documents are available at official webpage [37].

Following documents are available:

- Theory manual

- User guide for input syntax description
- Run examples
- Tutorial
- Installation and running guide
- SQA documents
- V&V suites

System Requirements

BISON is compatible with Linux and MacOS.

Most preferable operating system is Linux and MacOS environment. Minimum system requirements for MOOSE-based tools are:

- GCC/Clang C++14 compliant compiler (GCC @ 5.1.0, Clang @ 3.5.1 or greater)
- Memory: 16 GBs (debug builds)
- Processor: 64-bit x86
- Disk: 30GB

It is noted that Intel compilers are not supported.

Various Linux OS are compatible: CentOS, Fedora, OpenSUSE and Debian (Ubuntu, Mint).

MOOSE-based codes cannot be installed directly in Windows systems, hence, a user would need to install Linux environment under Windows such as Ubuntu 20.04. VcXrv is Windows X server which is necessary to use Peacock GUI.

BISON also allows installation and use under the INL HPC system.

Official computer OS comparison report is available upon request.

Easy Installation

Installation, update and uninstallation guideline for different OSes is described on the BISON webpage. Installation of MOOSE libraries is necessary prior to install BISON. The software can be pulled from the GitHub repository. No specific issues were found in installation.

Running syntax and tutorial with sample files are included in the code.

Graphic user Interface

BISON needs two separate input files to run the code: input text file and input mesh file based on Exodus-II finite element binary data format. The mesh generation external software compatible with BISON are: CUBIT, ABAQUS, PATRAN and ANSYS. For the post-processing and visualization of result, BISON is compatible ParaView, Ensight, Peacock, Blot, Patran, and Visit. Figure 4-4 shows an example of mesh generation with CUBIT and visualization of the result.

BISON also can use MOOSE's VectorPostprocessor system to generate vector of scalar values at each time step for post-processing especially to analyze fuel and cladding radial profilometry, radial temperature and burnup profiles.

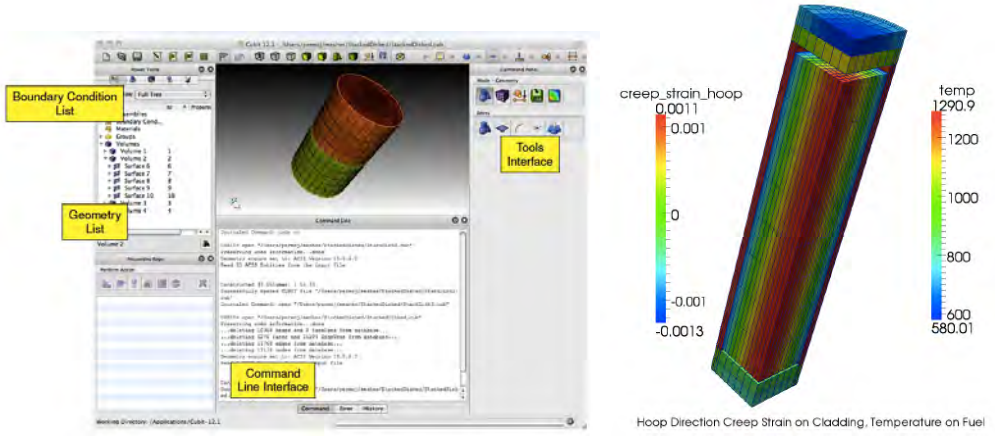

Figure 4-4: Mesh generation with CUBIT and post-processing example of BISON

Version Control

The BISON software is kept in a GitLab version control repository at INL. The MOOSE software is open-source and maintained in a similar version control repository know as GitHub. The BISON unit and regression tests, example problems, validation cases and documentation are maintained in the same BISON GitLab repository. This enables traceability of each code, validation case or documentation change.

Official version comparison report is available upon request.

V&V Activity/History

Verification activities are well-documented on the webpage. The list of verification and validation activities are shown in Section 0. Reports, papers, and history of BISON V&V are recorded on the webpage.

It is however recommended to provide a list of the regression tests and verification suites on the webpage.

QA Program

Quality of BISON is controlled under the SQA program of INL, which is compliance with ASME NQA-1. In INL, entire MOOSE-based tools follow same quality standard document “Software Quality Assurance Program of MOOSE and MOOSE-based Tools” (internal document). This activity includes documentation platform MooseDocs.

MOOSE and MOOSE-based applications are designated as QL-2 non-safety software but are managed to implement QL-1 safety software requirements. Generally, seismic analysis and risk assessment software are required to meet NQA-1 standards in order to be used for the design and risk assessment of several safety-critical facilities.

Following SQA documents are available.

- Software Quality Assurance Plan for MOOSE and MOOSE-based applications, including
  - Configuration Management Plan (CMP)
  - Verification and Validation Plan (VVP)
  - Software Quality Plan (SQP)
- Enterprise Architecture (EA) Record
- Safety Software Determination (SSD)

- Quality Level Determination (QLD)
- Software Design Description (SDD)
- System Requirement Specification (SRS) and Requirement Traceability Matrix (RTM)

Software design descriptions, requirement specifications on code functions and corresponding RTM is recorded in GitLab repository.

As of September 2021, no independent AMP is available. AMP MOOSE-based is currently not active.

Webpage

The official webpage is https://mooseframework.inl.gov/bison. This site includes basic information of the code, manuals, installation guides, examples and contact information for permission to use. All INL controlled MOOSE-based tools are in same host directory.

GitLab repository for BISON is https://hpcgitlab.hpc.inl.gov/idaholab/bison. It is noted that the access to the webpage needs a request and prior approval by the INL.

User Support

- BISON development team uses "Merge Request" system for use issue report under official GitLab repository.
- Frequently Asked Questions is under construction at BISON webpage.

Training Program

No regular training program is available. Training is available upon request.

Examples of BISON runs are available at webpage with input files in GitLab repository. Each example has problem, model, input, output and result description. Example cases are available for 1D, 2D and 3D cases of LWR fuel performance analysis, dimension switch, advanced fuels (TRISO, metal, MOX and ATF), tensor mechanics, and mortar migration.

License

BISON needs a license agreement with the INL. More information is available at the Nuclear Computational Resource Center (NCRC) webpage https://inl.gov/ncrc/.

4.2.5 Conclusions and Remarks

BISON is one of the most mature tools among many of the MOOSE-based tools. It has a wide range of V&V activities and its capabilities are demonstrated for assessing not only conventional LWR fuels but also fuels for advanced nuclear systems. Recently, BISON is expanding its capability on simulating ATF including performance assessment of U_3Si_2 fuel to support experiment, and pile creep and rupture test of FeCrAl cladding. In terms of code capabilities, following ATFs are identified for simulation with of BISON.

- FeCrAl cladding
- CrZr cladding
- U_3Si_2 fuel
- Cr_2O_3-doped UO_2 fuel
- Nitride and carbide fuels
- SiC/SiC cladding

It is however noted that experimental data for ATF are very limited to validate the code.

As a RISA Toolkit, BISON can provide great benefits for analyses related to fuel performance. It has a very high level of technical maturity and V&V activities. In terms of the RISA Pathway activities, no specific remarks are addressed. Technical maturity assessment results are shown in Table 4-3.

Table 4-3: BISON technology maturity assessment result

Requirements	Importance	Description	TRL
Development level	High	Code development is completed. Enhancement of related physical models are planned. GUI and pre- and post-processing methodologies are also set. Manuals, SQA and related documents are ready.	8
Use of proven technology	High	Mature technologies used in BISON. No specific issue was found.	9
PRA capability/applicability	Low	BISON has built-in sampling methods. Stochastic module in MOOSE can be used. RAVEN has tested for coupling.	6
Documentation	High	Manual or other supporting documents are available in official webpage. However, some part of webpage remains incomplete.	5
System requirements	Low	Windows version is not fully compatible, but no issue for running the code. Suggest performance test study in different OS.	7
Easy installation	Medium	Installation used GitLab pulling. MOOSE libraries need to be pre-installed. No specific issue was found.	9
Graphic user interface	High	Several of GUI software is available and tested. Peacock need performance test. No specific issue was found.	8
Version control	Low	GitLab maintains version. No specific version control report is available.	5
V&V activity/history	High	List of verification needs to be updated. Validation activities are fully performed.	8
QA program	Medium	SQA documents are available. Some of QL-1 documents are also available.	8
Webpage	High	Official webpage at MOOSE framework. However, some part of webpage remains incomplete.	5
User support	High	Developer supports directly to the issue through GitLab.	9
Training program	Low	Development team organizes training program as needed. Training tutorials and examples are available with documentation.	9
License	Low	License is controlled by INL.	9

4.3 GRIZZLY

4.3.1 Overview

Grizzly is a finite element-based simulation tool that can be applied to study a variety of degradation mechanisms in nuclear power plant components focused on the embrittlement of reactor pressure vessel (RPV) and the degradation of concrete structures. Grizzly models degradation process, but code also can evaluate performance of the degraded components. In general, RPV contains flaws generated from manufacturing process and can induce fracture nucleation sites during its lifetime. Due to embrittlement

caused from irradiation and high temperatures in NPP operation, as well as stresses caused by transient loading conditions, an RPV can be fractured during a severe accident. Grizzly has capabilities for performing engineering fracture assessments of RPVs. Grizzly also aims to model evolution of the microstructure and engineering properties of RPV for improved prediction of material embrittlement. For concrete structures, Grizzly has capability to model alkali-silica reaction (ASR) and radiation-induced volumetric expansion, and their effects on the mechanical response of concrete structures [50].

Developed by INL in partnership with other national laboratories, Grizzly was initially developed as part of the LWRS Program. The first applications were reported in 2012 with demonstration of RPV pressurized thermal shock [51]. In 2015, an initial demonstration was performed for use of Grizzly with RAVEN to study probabilistic assessment on flaw distribution in RPV [52]. Since that time, a standalone capability with Grizzly was developed and demonstrated for the probabilistic fracture mechanism analysis of RPVs [53]. Since Grizzly has a limited access policy, a MOOSE-based open-source structural material degradation simulation tool, BlackBear, was recently developed to provide wider access to the models not specific to nuclear applications to foster collaboration in this field [54].

4.3.2 Features

Grizzly uses MOOSE to solve fully coupled nonlinear partial differential equations such as thermal and mechanical response of RPVs and thermal, moisture, and mechanical models in concrete structures. The JFNK method is used for solving parallel nonlinear systems, and naturally supports effective coupling between physics equation systems modeled in MOOSE. MOOSE provides a comprehensive set of finite element support capability libraries (e.g., LibMesh) and provides for mesh adaptation and parallel execution. Grizzly also uses MOOSE embedded nonlinear solver PETSc. Major feature of Grizzly is assessment of system integrity by simulation of reactor metals (RPV and core internals) degradation mechanism such as embrittlement, fatigue, and corrosion induced from radioactive and high temperature environment.

Grizzly has an ability to perform probabilistic analysis on a population of flaws based on a 3D model of the global response of the RPV. This allows for local effects to be addressed in ways that are currently not possible with other codes. Grizzly provides a flexible framework for solution of the coupled physics models essential for modeling degradation mechanisms.

As a subset of Grizzly, BlackBear is an open-source MOOSE-based finite element method tool for modeling degradation phenomena in materials such as concrete and steel used in civil structures, as well as the response of those structures to the loading conditions that they are expected to safely withstand. More information is available at the Blackbear webpage (https://mooseframework.inl.gov/blackbear/).

4.3.3 Verification and Validation Status

As of September 2021, there are limited activities related to Grizzly V&V. Main physics from MOOSE kernels are verified in many other MOOSE-based tools such as BISON, which also uses same geometry and numerical approaches. Hence, Grizzly (and BlackBear) recently developed set of examples simulation and benchmark assessment test suites [55]. Currently, the Grizzly example problems consist of a set of models that compute the thermal/mechanical response of a prototypical RPV under transient loading conditions. These models all represent the same RPV and transient conditions, but with varying dimensionality:

- 1D axisymmetric
- 2D axisymmetric, with a single strip of elements
- 2D planar (cross-section of the RPV in a plane normal to the axis of rotation)
- 3D quarter symmetry of a cylinder (RPV beltline region only)
- 3D quarter symmetry model of full vessel

Following test cases are developed for engineering-scale RPV benchmarking.

- Global RPV models
- RPV probabilistic fracture mechanics models
- Initiation, growth and arrest model testing
- Plume analysis demonstration

Validation suites for ASR in concrete are:

- Concrete block cases
- Reinforced concrete beam member
- Concrete modeling

4.3.4 Technical Maturity Assessment

Following information describes Grizzly technical maturity assessment results based on the requirements as the RISA Toolkit.

Development Level

Though the development was started in early 2010's, Grizzly code is still under development especially for validation and demonstration of the code. Due to export control regulations on tools for simulating reactor components, the code access and simulation results are export controlled.

Similar to other MOOSE-based tools, Grizzly follows general features of MOOSE framework such as coupling flexibility, GUI, parallel execution, etc.

Official webpage exists but it is not included on the new MOOSE framework webpage (https://mooseframework.inl.gov/). The Grizzly webpage has limited information. Only user manual is available.

The Grizzly input file can be generated either using text editors or MOOSE-based GUI tool Peacock, and the mesh can be generated using built-in tools and external software such as Cubit.

Grizzly can be installed on standalone computers or on HPC systems. Users need a licensing agreement with INL to access this code.

Use of Proven Technology

Grizzly uses MOOSE kernels to solve relevant physics. The PETSc nonlinear solver package and LibMesh are used for the finite element discretization. These libraries allow use PJFNK method to maximize computational effectiveness for solving nonlinear analysis. For linear analysis Newton method is used to ease convergence. These technologies are mature and fully verified and validated.

Supporting software for post- and pre-processing are also showing high level of maturity. However, performance of the Peacock has not been evaluated under the RISA Pathway project.

PRA Capability/Applicability

Grizzly has been coupled with RAVEN to simulate probabilistic analysis of behavior of pre-existing flaws in RPV and fracture model mechanics [53]. Similar to BISON, Grizzly also can use MOOSE stochastic tool system.

Documentation

Grizzly has an automated system that uses a combination of information in the source code and Markdown files to generate documentation that is consistent with the current state of the code. This is hosted on an internal INL website and is available to users with access to the code.

Following documents are available:

User manual [56]

System Requirements

Grizzly is compatible with Linux and MacOS.

Most preferable operating system is Linux and MacOS environment. Minimum system requirements for MOOSE-based tools are:

- GCC/Clang C++14 compliant compiler (GCC @ 5.1.0, Clang @ 3.5.1 or greater)
- Memory: 16 GBs (debug builds)
- Processor: 64-bit x86
- Disk: 30GB

It is noted that Intel compilers are not supported.

Various Linux OS are compatible: CentOS, Fedora, OpenSUSE and Debian (Ubuntu, Mint).

MOOSE-based codes cannot be installed directly in Windows systems; hence, a user would need to install Linux environment under Windows such as Ubuntu 20.04. VcXrv is Windows X server which is necessary to use Peacock GUI.

Grizzly also allows installation and use under the INL HPC system.

Official computer OS comparison report is available upon request.

Easy Installation

Similar to other MOOSE-based tools, Grizzly uses GitLab system constructed under INL HPC system.

Graphic user Interface

Grizzly needs two separate input files to run the code: input text file and input mesh file based on Exodus-II finite element binary data format. Peacock can generate the mesh and output graphics. ParaView is also compatible for output visualization. Figure 4-5 is example of Grizzly mesh and simulation results.

Grizzly also can use VectorPostprocessor system of MOOSE to generate vector of scalar values at each time step for post-processing.

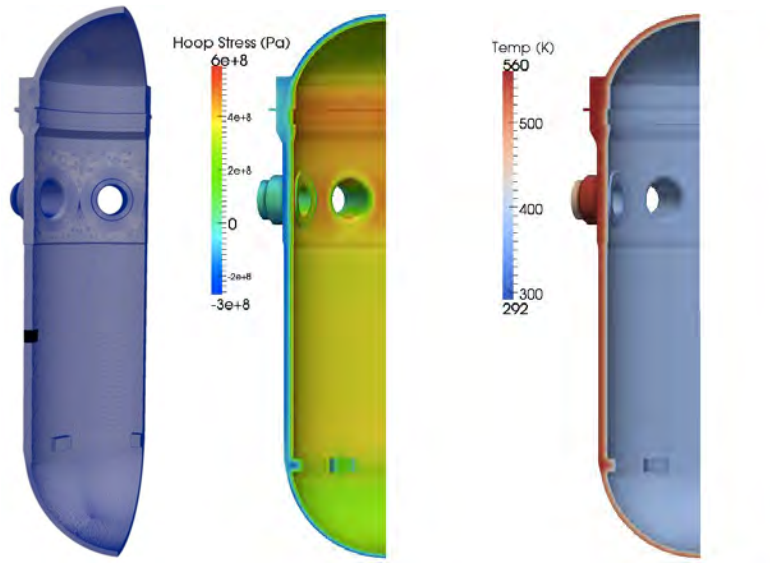

Figure 4-5: Mesh and post-processed output example of Grizzly

Version Control

Similar to BISON, the Grizzly software is kept in a GitLab version control repository at INL. The Grizzly unit and regression tests, example problems, validation cases and documentation are maintained in the same Grizzly GitLab repository. This enables traceability of each code, validation case or documentation change.

Official version comparison report is available upon request.

V&V Activity/History

As of September 2021, two validation studies have been performed. However, the report is not publicly accessible available. One benchmark report is available for basic V&V purpose [55].

QA Program

Quality of Grizzly is controlled under the INL's SQA program, which is in compliance with ASME NQA-1. All MOOSE-based tools follow the same quality standard document "Software Quality Assurance Program of MOOSE and MOOSE-based Tools" (internal document). This activity includes documentation platform MooseDocs.

MOOSE and MOOSE-based applications are designated as QL-2 non-safety software but are managed to implement QL-1 safety software requirements. Generally, risk assessment software are required to meet NQA-1 standards in order to be used for the design and risk assessment of nuclear safety-critical facilities.

Following SQA documents are available.

- Software Quality Assurance Plan for MOOSE and MOOSE-based applications, including
  - Configuration Management Plan (CMP)
  - Verification and Validation Plan (VVP)
  - Software Quality Plan (SQP)
- Enterprise Architecture (EA) Record
- Safety Software Determination (SSD)
- Quality Level Determination (QLD)

- Software Design Description (SDD)
- System Requirement Specification (SRS) and Requirement Traceability Matrix (RTM)

Software design descriptions, requirement specifications on code functions and corresponding RTM is recorded in GitLab repository.

As of September 2021, no independent AMP is available. AMP MOOSE-based is currently not active.

Webpage

The official webpage is https://moose.inl.gov/grizzly/SitePages/Home.aspx. This site includes basic information of the code, manual, list of publication and contact information.

User Support

Mailing list and web-based discussion forum is provided for licensed Grizzly users.

Training Program

No regular training program is available. Training is available upon request.

License

Grizzly needs a license agreement with the INL. More information is available at the NCRC webpage https://inl.gov/ncrc/.

4.3.5 Conclusions and Remarks

Grizzly is still under development under the LWRS Materials Pathway research area. The recent interest on subsequent license renewal (SLR) of LWRs up to 80 years brings more interest on the use of Grizzly for expanded material degradation assessments. Major fields of interest include core internals and piping, RPV steel, concrete structures, and electrical cabling and insulation. As a RISA Toolkit, Grizzly can be used for risk-informed approach for SLR assessments. Technical maturity assessment results are shown in Table 4-4.

Table 4-4: Grizzly technology maturity assessment result

Requirements	Importance	Description	TRL
Development level	High	Code is still under development. GUI and pre- and post-processing methodologies are ready. Theory manual and SQA related documents are ready.	5
Use of proven technology	High	Mature technologies used in Grizzly. No specific issue was found.	9
PRA capability/applicability	Low	Grizzly can be coupled with RAVEN, and has been demonstrated.	5
Documentation	High	Manual or other supporting documents are not fully available in official webpage.	3
System requirements	Low	Windows version is not fully compatible, but no issue for running the code. Suggest performance test study in different OS.	7
Easy installation	Medium	Installation uses GitLab pulling. MOOSE libraries need to be pre-installed. No specific issue was found.	9
Graphic user interface	High	Peacock and ParaView are compatible. No specific issue was found.	8
Version control	Low	GitLab maintains version. No specific version control report is available.	5
V&V activity/history	High	No specific V&V report is available. Benchmark study was	3

Requirements	Importance	Description	TRL
		performed for V&V purpose.	
QA program	Medium	SQA documents are available. Some of QL-1 documents are also available.	8
Webpage	High	Not many information is available at webpage. However, licensed users can access to GitLab repository with more available information.	3
User support	High	No information.	N/A
Training program	Low	No information.	N/A
License	Low	License is controlled by INL.	9

REFERENCES

[1] S. Lawrence, et al., Risk-Informed Systems Analysis (RISA) Pathway Technical Program Plan, INL/EXT-21-01601, Idaho National Laboratory, 2021

[2] Y-J Choi, Assessment of Verification and Validation Status - RELAP5-3D and RAVEN, INL/EXT-19-56151, Idaho National Laboratory, 2019

[3] Y-J Choi, Assessment of Verification and Validation Status - EMRALD and HUNTER, INL/EXT-20-59904, Idaho National Laboratory, 2020

[4] C. Smith et al., RELAP-7 Software Verification and Validation Plan Requirements Traceability Matrix (RTM) Part 1 – Physics and numerical methods, INL/EXT-15-36684, Idaho National Laboratory, 2015

[5] J. Yoo, Y-J Choi, Advanced Validation Risk-Informed Approach for the Flooding Tool Based Upon Smooth Particle Hydrodynamics - Validation and Development Status of NEUTRINO, INL/EXT-18-44976, Idaho National Laboratory, 2018

[6] W. Oberkampf, T. Trucano, Verification and Validation in Computational Fluid Dynamics, Progress in Aerospace Science, Vol. 38, pp.209-272, 2002

[7] W. Oberkampf, T. Trucano, C. Hirsch, Verification, validation, and predictive capability in computational engineering and physics, Applied Mechanics Reviews, Vol. 57, No. 5, pp.345-384, 2004

[8] J. Ferziger, M. Peric, Further discussion of numerical errors in CFD. Int. J. Numerical Methods Fluids, Vol. 23, pp.1263-1274, 1996

[9] I. Babuska, et al., A posteriori estimation and adaptive control of the pollution error in the h-version of the finite element method, Int. J. Numerical Methods Engineering, Vol. 38, pp.4207-4235, 1995

[10] L. Hatton, The T experiments: errors in scientific software, IEEE Computational Science and Engineering, Vol. 4(2), pp. 27-38, 1997

[11] D. Haworth et al., A global approach to error estimation and physical diagnostics in multidimensional computational fluid dynamics, Int. J. of Numerical Methods Fluids, Vol. 17(1), pp. 75–97, 1993.

[12] J. Coleman, et al., Plan to Verify and Validate Multi-Hazard Risk-Informed Margin Management Methods and Tools, INL/EXT-16-39195, Idaho National Laboratory, 2016

[13] C. Roy, W. Oberkampf, A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing, Computational Methods in Applied Mechanics and Engineering, Vol. 200, pp. 2131-2144, 2011

[14] W. Oberkampf, T. Trucano, Verification and validation benchmarks, Nuclear Engineering and Design, Vol. 238, pp. 716-743, 2008

[15] C. Parisi, Y-J Choi, Risk-Informed Multi-Physics Best-Estimate Plus Uncertainties (BEPU) Application Development of RELAP5-3D Perturbation Model, INL/EXT-20-59594, Idaho National Laboratory, 2020

[16] IEEE Computer Society, IEEE Standard for System, Software, and Hardware Verification and Validation, IEEE Std. 1012-2016, 2016

[17] U.S. NRC, Verification, validation, reviews, and audits for digital computer software used in safety systems of nuclear power plants, Regulatory Guide 1.168, Revision 2, 2013.

[18] DOE, DOE O 414.1C (Quality Assurance), U.S. DOE, 2005

[19] DOE, DOE G 414.1-4 (Safety Software Guide for Use with 10 CFR 830 Subpart A, Quality Assurance), U.S. DOE, 2010

[20] ASME, Quality Assurance Requirements for Nuclear Facility Applications, NQA-1-2019, 2019

[21] https://mooseframework.inl.gov/mastodon/

[22] R. Szilard, et al., RISMC Toolkit and Methodology Research and Development Plan for External Hazards Analysis, INL/EXT-16-38089, Idaho National Laboratory, 2016

[23] J. Coleman, et al., Development Plan for the External Hazards Experimental Group, INL/EXT-16-38328, Idaho National Laboratory, 2016

[24] J. Coleman, et al., Plan to Verify and Validate Multi-Hazard Risk-Informed Margin Management Methods and Tools, INL/EXT-16-39195, Idaho National Laboratory, 2016

[25] C. Bolisetti, et al., Advanced Seismic Probabilistic Risk Assessment Methodology: Development of Beta 1.0 MASTODON Toolset, INL/EXT-17-43148, Idaho National Laboratory, 2017

[26] A. Kammerer, et al., Nonlinear Soil-Structure-Interaction Analysis in Support of Seismic Design and Probabilistic Risk Assessment of Nuclear Facilities, INL/EXT-18-50155, Idaho National Laboratory, 2018

[36] S. Veeraghavan, et al., MASTODON: An Open-Source Software for Seismic Analysis and Risk Assessment of Critical Infrastructure, Nuclear Technology, Vol. 27, pp. 1073-1095, 2021

[37] https://mooseframework.inl.gov/bison/index.html

[38] R. Williamson et al., Enhancing the ABAQUS thermomechanics code to simulate multipellet steady and transient LWR fuel rod behavior, J. of Nuclear Materials, Vol. 415, pp. 74-83, 2011

[39] R. Williamson et al., Multidimensional muliphysics simulation of nuclear fuel behavior, J. of Nuclear Materials, Vol 423, pp. 149-163, 2012

[40] C. Smith et al., Accident Tolerant Fuel Analysis, INL/EXT-14-33200, Idaho National Laboratory, 2014

[41] K. Gamble et al., BISON and MARMORT Development for Modeling Fast Reactor Fuel Performance, Fuel Cycle Research & Development Advanced Fuel Campaign, Idaho National Laboratory, INL/EXT-15-36440, 2015

[42] H. Zhang et al., Risk-Informed ATF and FLEX Analysis for an Enhanced Resilient BWR Under Design Basis and Beyond-Design-Basis Accidents, INL/EXT-20-59906, Idaho National Laboratory, 2020

[43] J. Hales et al., Verification of the BISON fuel performance code, Annals of Nuclear Energy, Vol. 71, pp. 81-90, 2014

[44] A. Toptan et al., FY20 Verification of BISON using analytic and manufactured solutions, U.S. DOE, CASL-U-2020-1939-000, 2020

[45] R. Williamson et al., Validating the BISON fuel performance code to integral LWR experiments, Nuclear Engineering and Design, Vol. 301, pp. 232-244, 2016

[49] C. Rabiti et al., RAVEN User Manual, Idaho National Laboratory, INL/EXT-15-34123 Rev. 7, 2021

[50] https://moose.inl.gov/grizzly.

[51] B. Spencer et al., A Proof of Concept: Grizzly, the LWRS Program Materials Aging and Degradation Pathway Main Simulation Tool, Idaho National Laboratory, INL/EXT-12-27559, 2012

[52] B. Spencer at al., Initial Probabilistic Evaluation of Reactor Pressure Vessel Fracture with Grizzly and RAVEN, Idaho National Laboratory, INL/EXT-15-37121, 2015

[53] B. Spencer at al., Probabilistic Fracture Mechanics of Reactor Pressure Vessels with Populations of Flaws, Idaho National Laboratory, INL/EXT-16-40050, 2016

[54] B. Spencer at al., Grizzly and BlackBear: Structural Component Aging Simulation Codes, Nuclear Technology, Vol. 207, pp. 981-1003, 2021

[55] B. Spencer at al., Assessment of Grizzly Capabilities for Reactor Pressure Vessels and Reinforced Concrete Structures, Idaho National Laboratory, INL/EXT-20-00617, 2020

[56] B. Spencer at al., Grizzly Usage and Theory Manual: Version 1.0 Beta, Idaho National Laboratory, INL/EXT-16-38310, 2016